МЕЃУНАРОДЕН СЛАВЈАНСКИ УНИВЕРЗИТЕТ
„ГАВРИЛО РОМАНОВИЧ ДЕРЖАВИН“
СВЕТИ НИКОЛЕ
РЕПУБЛИКА СЕВЕРНА МАКЕДОНИЈА



ФАКУЛТЕТ ЗА ИНФОРМАТИКА
СТУДИСКА ПРОГРАМА: СОФТВЕРСКИ ТЕХНОЛОГИИ

МАГИСТЕРСКА РАБОТА

РАЗВОЈ НА ИГРИ СО UNITY: ДВОДЕЛНО ПАТУВАЊЕ КОН СОЗДАВАЊЕ И ИМПЛЕМЕНТАЦИЈА

СТУДЕНТ: ЈАНЕ ЉУЗОВ
БРОЈ НА ИНДЕКС: M219011
МЕНТОР: ПРОФ. Д-Р МИМОЗА А. ЈАНКУЛОВСКА

Битола, 2023

АПСТРАКТ

Овај труд се фокусира на принципите на развој на игри и улогата на Unity во индустријата. Трудот има за цел да го истражи пресекот на теоријата и практичната примена, истражувајќи ја теоријата за дизајн на игри, психологијата на играчите, техниките на раскажување на приказни, механиката на игрите и моделите на дизајнирање. Со анализа на постоечките истражувања и најдобри практики, како и проучување на карактеристиките и способностите на Unity, ова истражување се обидува да обезбеди увид во тоа како Unity го олеснува спроведувањето на принципите за развој на игри. Трудот вклучува и развој на бета игра со користење на Unity, документирање на процесот, предизвиците и направените подобрувања. Теоретското и практичното значење на ова истражување лежи во неговиот придонес во областа на развојот на игри, обезбедувајќи подлабоко разбирање за успешно креирање на игра и подобрување на целокупното искуство на играчите.

Клучни зборови:

Развој на игри, Unity, теорија на дизајн на игри, психологија на играчите, техники на раскажување приказни, механика на игри, модели на дизајнирање, практична примена, проект за бета игра, оптимизација, искуство на играчи.

СОДРЖИНА

АПСТРАКТ .....

Клучни зборови: .....

ВОВЕД 1

I. ТЕОРЕТСКИ ДЕЛ 4

1 ПРИНЦИПИ НА РАЗВОЈ НА ИГРИ 4

1.1 Теорија на дизајн на игри 4

1.2 Психологија на играчите во игрите 10

1.3 Техники на раскажување на приказните во игрите 20

1.4 Анализа на механиката на игрите 26

1.5 Шаблони на дизајн во развојот на игри 33

II. ИСТРАЖУВАЊЕ НА UNITY 39

2 ЗАПОЗНАВАЊЕ СО UNITY 39

2.1 Unity 39

2.1.1 Интерфејс на Unity 40

2.1.2 Основни карактеристики во Unity 44

2.1.3 Управување со средства (Assets) во Unity 47

2.1.4 Композиција на сцена во Unity 55

2.1.5 Скриптирање и кодирање во Unity 84

III. АНАЛИЗА НА ИСТРАЖУВАЊЕТО, МЕТОДОЛОГИЈА И ДИСКУСИЈА 96

3 ИСТРАЖУВАЊЕТО И ИНТЕГРИРАЊЕТО НА ПРИНЦИПИТЕ ЗА РАЗВОЈ НА ИГРИ СО UNITY 96

3.1 Методологија на преглед на литература 96

3.2 Примена на теоријата за дизајн на игри во Unity 99

3.3 Инкорпорирање на психологијата на играчите во игрите 100

3.4 Дискусија 103

IV. БЕТА- ИГРА 105

4 КОНЦЕПТУАЛИЗАЦИЈА И ДИЗАЈН 105

4.1 Концепт на играта и механика 107

4.2 Размислувања за дизајн и кориснички интерфејс 109

4.3 Имплементација на играта со помош на Unity 110

4.4 Тестирање 114

4.5 Идни планови 114

ЗАКЛУЧОК 115

КОРИСТЕНА ЛИТЕРАТУРА 117

Прилог 1: Слики од играта 126

Прилог 2: Дел од Скрипти 127

ЛИСТА НА СЛИКИ

1- 1 Теорија на дизајн на игри 4
1- 2 дијаграм што споредува „дизајн на ниво“ наспроти „уметност на животната средина“, користејќи слики од процесот на Истражувачкиот објект Галенте од Dust 514 7
1- 3 Модел на мотивација на играч 11
1- 4 Модел на проток поврзан со нивоата на предизвикот и вештината. 12
1- 5 Модел на мотивација според теоријата на Бартл (1996). 13
2- 1 Лого на Unity 39
2- 2 Стандарден кориснички интерфејс на Unity 40
2- 3 Поглед на сцена (Scene window). 41
2- 4 Поглед на играта (Game view). 41
2- 5 Прозорец за инспектор на објектот на главната камера. 42
2- 6 Прозорецот хиерархија. 42
2- 7 Прозорец на проектот. 43
2- 8 Прозорецот конзола. 43
2- 9 Креирање на празен GameObject. 44
2- 10 New Sprite GameObject. 44
2- 11 Додавање компонента во GameObject. 45
2- 12 Организирање на проект. 45
2- 13 Лента со алатки 46
2- 14 Работен тек на средствата во Unity 47
2- 15 Прозорецот Проект ги прикажува средствата што Unity ги внесува во вашиот проект 50
2- 16 Со кликнување на средството со слика во прозорецот Проект се прикажуваат поставките за внес за тоа средство во Инспекторот 51
2- 17 Аудио средство избрано во прозорецот Проект ги прикажува поставките за внес на аудио за тоа средство во Инспекторот 51
2- 18 Прозорец за изнес на пакет 54
2- 19 Нова празна сцена, со стандардни 3D објекти - камера и насочено светло. 56
2- 20 The New Scene dialog 57
2- 21 Закачување на шаблон 58
2- 22 Лоцирање и уредување на шаблон 58

2-23 Зачуваните средства на сцената видливи во прозорецот на проектот 62
2-24 Истата сцена прикажана во режим на перспектива (лево) и ортогонален режим (десно) 63
2-25 Ефект на осветлено место во сцената 65
2-26 Сцена со предмети кои фрлаат сенки 67
2-27 Типичен приказ на машина за состојба на анимација во прозорецот на аниматорот 68
2-28 Дијаграм што покажува како различни делови од системот за анимација се поврзуваат заедно 70
2-29 Секцијата за проширување на криви на картичката за анимација 71
2-30 Кривите на внесениот клип за анимација 72
2-31 Уредувач на кривата во Unity 72
2-32 Прозорецот Аниматор покажува график на целото стебло на мешање. Лево е стебло на мешање со само коренот јазол за мешање (сè уште не се додадени детски јазли). Десно е стеблото на мешање со коренот јазол за мешање и три анимациски клипови како детски јазли. 75
2-33 Јазол за мешање прикажан во инспекторот пред да се додадат какви било движења. Иконата плус се користи за додавање анимациски клипови или деца на стеблото на мешање. 75
2-34 Контексното мени кога се кликнува со десното копче на јазолот од стеблото на мешање. 76
2-35 2D Стебло на мешање поставено со пет анимациски клипови, кои се прегледуваат во инспекторот 76
2-36 Пример за транзиција како што се гледа во инспекторот. 77
2-37 Именување на транзиции 77
2-38 Транзициите што состојбата ги користи 78
2-39 Поглед на својствата на транзицијата 79
2-40 Поставките за транзиција и графикот како што е прикажано во Инспекторот 82
2-41 Контролите за преглед на параметарот стебло на мешање, видливи кога вашата тековна или следната состојба е состојба на стебло на мешање. 83
2-42 Лого на C# 86
2-43 Game Object со прикачена компонента Rigidbody 87
2-44 Прелистувач на Компоненти 88
2-45 Аудио изворот со звучен ефект е референциран правилно во аудио клипот 89

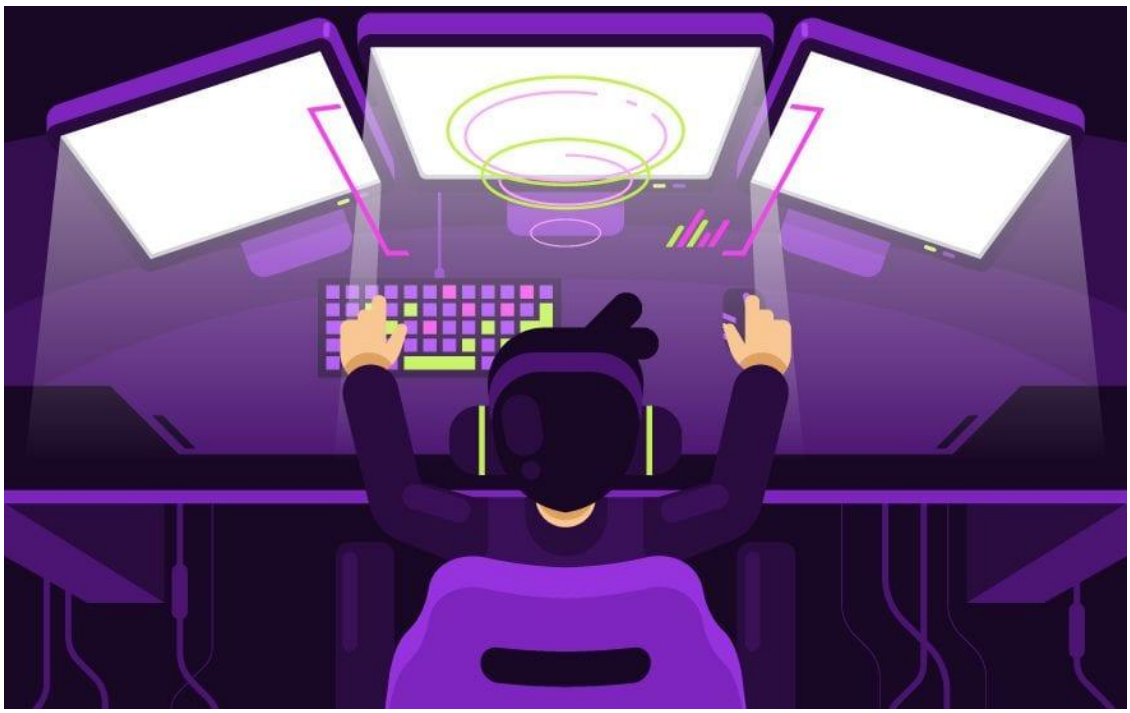
4- 1 Почетна концепт скица за карактер и пречки 105
4- 2 Трансформирање на скицата во полиран дизајн на карактерот и пречките со помош на Illustrator 106
4- 3 Ран дизајн на играта 107
4- 4 Приказ на мени-ата во играта 111
4- 5 Приказ на прозорецот кога играта ќе заврши и опциите за враќање во менито ли повторно играње 113

КОРИСТЕНИ КРАТЕНКИ

Кратенка	англиски	македонски
Fr	Frame rate	Стапка на слики
Erp	Electroencephalography	Електроенцефалографија
Xp	Experience	Поени за искуство
OOD	Object oriented design	Објектно ориентиран дизајн
VR	Virtual reality	Виртуелна реалност
AR	Augmented reality	Проширена реалност
FOV	Field of view	видно поле
fMRI	Functional magnetic resonance imaging	Функционална магнетна резонанца
API	Application Programming Interface	Програмски интерфејс за апликации
IDE	Integrated development environment	Интегрирана развојна средина
UI	User interface	Кориснички интерфејс
CPU	Central processing unit	Централната единица за обработка
GPU	Graphics processing unit	Единица за обработка на графика
NPC	Non-player character	Неиграчки лик

ВОВЕД

Развојот на игри еволуираше во динамично и брзо проширувачко поле, поттикнато од технолошкиот напредок и постојано растечката побарувачка за извонредни интерактивни искуства. Во ова поле, Unity, популарен погон за развој на игри, игра клучна улога, обезбедувајќи им на програмерите моќен пакет со алатки и разновидна платформа за да ги оживеат нивните креативни визии. Овој труд има за цел да ги истражи принципите на развој на игри и улогата на Unity во индустријата, истражувајќи го нивното вкрстување и влијание врз создавањето привлечни игри.



Проблемот е во потребата да се разберат основните аспекти на принципите за развој на игри и како тие се поврзани со практичната примена на Unity. Како што индустријата за игри продолжува да расте и да се развива, од суштинско значење е да се проучат основните теории, методологии и најдобри практики за ефикасно дизајнирање и развој на игри кои ги ангажираат играчите на повеќе нивоа. Дополнително, со оглед на тоа што Unity е широко прифатен од развивачите со различни нивоа на вештини, императив е да се истражи како овој погон го олеснува спроведувањето на овие принципи и ги овластува креаторите на игри да создаваат извонредни искуства.

Примарната цел на ова истражување е да се спроведе длабинско испитување на принципите за развој на игри и улогата на Unity во индустријата. Навлегувајќи во постоечката литература и најдобрите практики, се прави обид да се идентификуваат клучните аспекти и предизвици во теоријата на дизајн на игри, психологијата на играчите, техниките на раскажување приказни и механиката на игрите. Понатаму, целта е да се анализираат начините на кои Unity ги интегрира и поддржува овие принципи, нудејќи цврста рамка за развој на игри.

Ова истражување воведува сеопфатно истражување на темата, почнувајќи со преглед на литература кој опфаќа различни извори на информации, вклучувајќи академски трудови, публикации во индустријата и реномирани книги од областа на развојот на игри. Прегледот ќе обезбеди основа за разбирање на моменталната состојба на истражувањето и најдобрите практики, истакнувајќи ги празнините во знаењето и контроверзите кои бараат дополнително истражување.

Содржината на овој труд е структурирана на неколку различни делови. Првиот дел се фокусира на општите принципи за развој на игри, вклучувајќи теорија на дизајн на игри, психологија на играчите, техники на раскажување приказни, анализа на механиката на играта и шеми на дизајнирање. Вториот дел го истражува Unity, опфаќајќи го неговиот интерфејс, достапните алатки, системот за управување со средства, составот на сцената, можностите за анимација и скриптирањето во околината на Unity. За крај е развојот на бета играта со помош на Unity, при што се користи дел од знаењето стекнато од деловите еден и два. Процесот на развој е документиран, се дискутираат предизвиците и се собираат повратни информации за итеративно да се подобри проектот за играта.

Теоретското и практичното значење на ова истражување лежи во неговиот придонес во областа на развојот на игри. Со сеопфатно истражување на принципите и практиките на развој на игри и нивна интеграција во Unity, ова истражување има за цел да им обезбеди на програмерите, студентите и научниците подлабоко разбирање на факторите кои придонесуваат за успешно создавање на игра. Увидите добиени од ова истражување не само што ќе ги информираат идните напори за развој на игри, туку и ќе ја олеснат оптимизацијата на процесите на дизајнирање игри и ќе го подобрат целокупното искуство на играчите.

Накратко, овај труд има за цел да ги истражи принципите на развој на игри и улогата на Unity во индустријата. Преку длабински преглед на литературата, анализа на принципите за развој на игри, истражување на карактеристиките на Unity и развој на бета игра, цел е да се придонесе разбирање и унапредување на практиките за развој на игри. Преку испитување на пресекот на теоријата и практичната примена во рамките на Unity, ова истражување се обидува да ги поттикне развивачите на игри со знаења и вештини неопходни за создавање привлечни и извонредни искуства за играње.

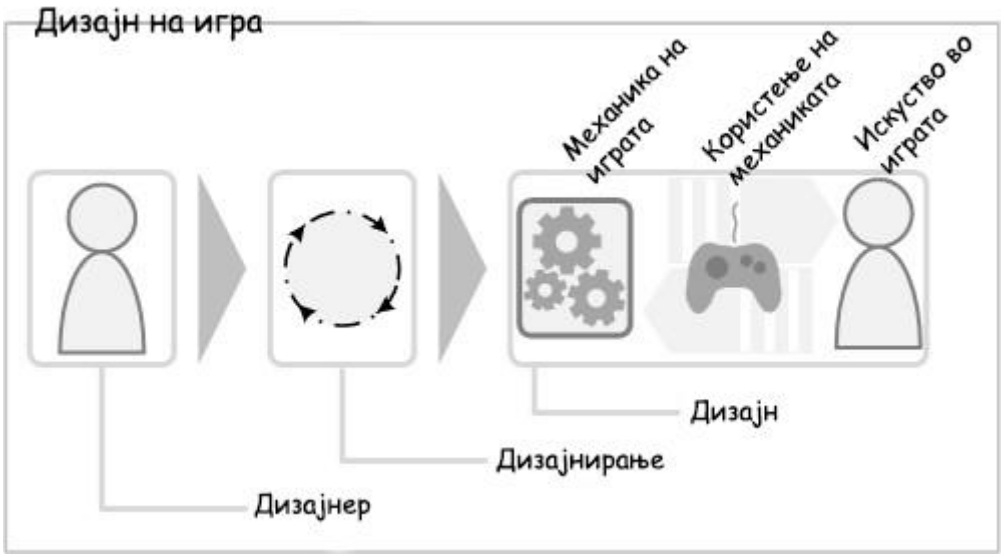
I. ТЕОРЕТСКИ ДЕЛ

1 ПРИНЦИПИ НА РАЗВОЈ НА ИГРИ

Овој дел обезбедува теоретска основа за разбирање на различните аспекти на развојот на играта.

1.1 Теорија на дизајн на игри

Овој поддел ги истражува принципите и теориите на дизајнот на играта. Опфаќа теми како што се механика на игри, дизајн на ниво и ангажман на играчите. Се дискутира за основните принципи на дизајнирање привлечни искуства за играње, вклучувајќи ја важноста на јасни цели, значајни избори и добро дефинирани системи за прогресија.



1- 1 Теорија на дизајн на игри

Механика на игри

Механиката на играта е суштински елемент кој дефинира како играта функционира и ги ангажира играчите. Механиката може да се измери со користење на прагови како што се стапката на слики (fr) и време на одговор за да се оцени контролата во реално време. Одржување на непречено движење со минимална стапка на слики и брзо време на одговор обезбедува беспрекорно и интерактивно искуство. Разложувањето на интеракциите на играта во индивидуални механики помага да се оцени нивното придржување до стандардите за контрола во реално време. [98, 99, 108, 121]

Механиката на играта ги претставува дејствата и способностите што играчите ги имаат во играта. Тие ја формираат основата на играта и можат да коегзистираат, еволуираат и да се менуваат во текот на играта, додавајќи длабочина и прогресија. Анализирањето на интеракцијата на механиката, контролата во реално време, симулираниот простор и полирањето помага да се оцени целокупното чувство на играта. [2, 22, 106]

Разбирањето на механиката на играта е од клучно значење за дизајнерите на игри бидејќи тие го обликуваат искуството на играчот. Вештина и шанса се важна механика во дизајнот на играта, предизвикувајќи ги вештините на играчите и воведувајќи несигурност. Други механики како опасности, реквизити и загатки создаваат предизвици и можности за донесување одлуки. [14, 78, 117]

Ефективната механика на играта прави баланс помеѓу да се биде интуитивен и нудење на длабочина. Интуитивната механика е лесна за разбирање и навигација, додека длабочината овозможува истражување, експериментирање и мајсторство. Агенцијата на играчите (Player Agency)[1], способноста за донесување значајни одлуки, е од суштинско значење за ангажирањето на играчите. [72, 90, 134]

Механиката на играта треба да се усогласи со целокупниот дизајн на играта и целната публика. Програмерите користат итеративен процес на дизајнирање, експериментирање со различни идеи, механика за прототип и собирање повратни информации од играчите за да го усовршат и подобрат искуството во играта. [115, 132]

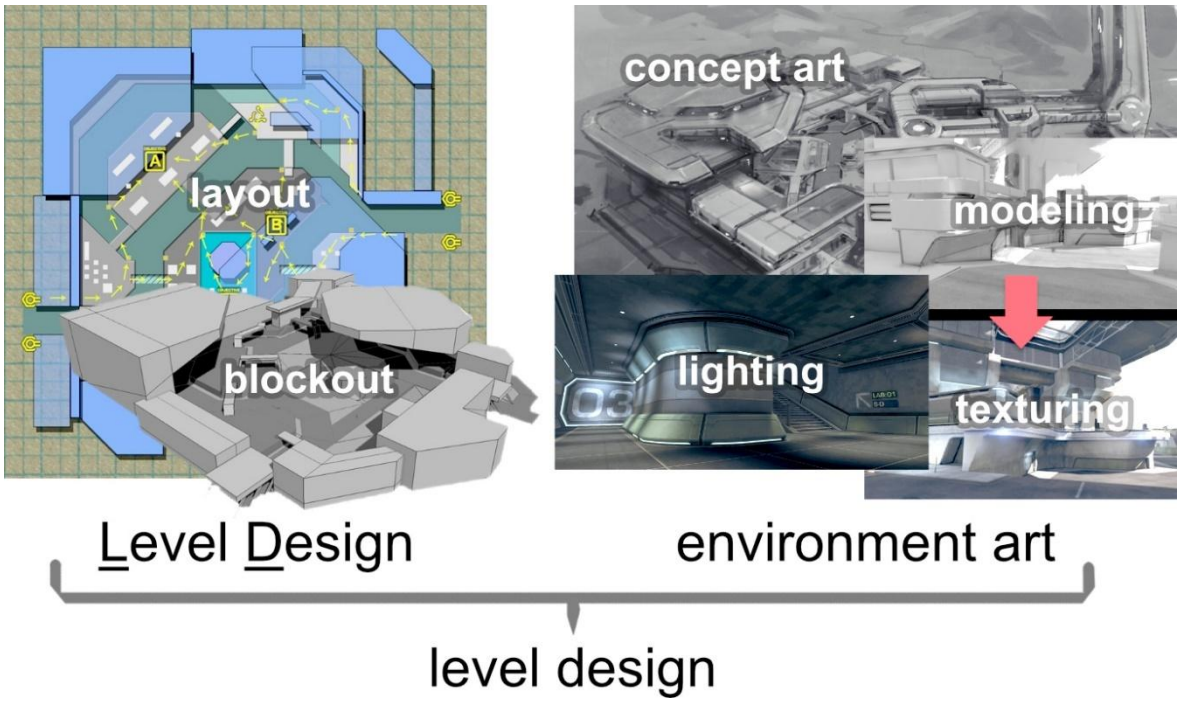
[1] Способноста на играчот да влијае на приказната преку дизајнот на играта или гејмплејот.

Со фокусирање на основниот сет на механика и воведување сателитска механика, дизајнерите на игри создаваат игри со едноставна, но привлечна игра. Овој пристап овозможува континуирана еволуција и владеење на основните активности, истовремено обезбедувајќи новина за одржување на мотивацијата на играчите. Исто така, ги насочува процесите на развој, тестирање и балансирање. [11, 21]

Накратко, механиката на играта е основата на однесувањето на играта и за искуството на играчите. Тие опфаќаат широк опсег на интеракции и можат да се категоризираат во различни типови. Дизајнирањето на интуитивна, длабока и усогласена механика резултира со извонредни и волшебни искуства во играта.

Дизајн на ниво

Дизајнот на нивоа во развојот на играта се фокусира на создавање привлечни и извонредни искуства за играчите во виртуелниот свет на играта. Вклучува внимателна изработка на средината, пречки, загатки и предизвици за да се обезбеди избалансиран и пријатен тек на играта. Неколку клучни концепти придонесуваат за ефективен дизајн на ниво. [127]



1- 2 дијаграм што споредува „дизајн на ниво“ наспроти „уметност на животната средина“, користејќи слики од процесот на Истражувачкиот објект Галенте од Dust 514

Прво, добро дизајнираните нивоа треба да имаат јасна и кохезивна структура, дозволувајќи им на играчите да се движат низ нив без конфузија. Визуелните знаци како што се патеките, знаменитостите и осветлувањето им помагаат да ги водат играчите и да ги наведат следните чекори. Мешавина од отворени простори и ограничени области додава разновидност и ја одржува играта привлечна. [127]

Второ, дизајнот на ниво треба да ја земе во предвид кривата на тежина. Постепеното воведување и зголемување на предизвиците им овозможува на играчите да учат и да се прилагодат на механиката на играта. Започнувањето со

премногу тешки задачи може да доведе до фрустрација, па затоа е важно да се обезбеди чувство на успех со постепено зголемување на сложеноста и тежината. [127]

Поттикнувањето на истражување и откривање е уште еден важен аспект на дизајнот на ниво. Скриените области, тајните патеки и изборните цели ги наградуваат љубопитните играчи и додаваат длабочина на искуството во играта. нелинеарниот дизајн со повеќе патеки ја подобрува можноста за повторување и нуди различни стратегии за завршување на нивото. [127]

Естетиката, игра клучна улога во дизајнот на ниво, придонесувајќи доживување во целокупното искуство. Вниманието на деталите, шемите на бои, осветлувањето и соодветната употреба на реквизити и комплети помагаат да се создаде посакуваната атмосфера. Визуелниот дизајн треба да се усогласи со наративот на играта и да ги предизвикува наменетите емоции кај играчите. [127]

Тестирањето и повторувањето се неопходни во дизајнот на ниво. Собирањето повратни информации од учесниците во играта помага да се идентификуваат областите на кои им треба подобрување. Анализирањето на однесувањето на играчите, скоковите на тежината и темпото на играта овозможуваат префинетост на дизајнот на нивоата. Итерацијата заснована на повратни информации од играчите гарантира дека нивото ги исполнува очекувањата на целната публика и обезбедува пријатно искуство. [127]

Накратко, дизајнот на ниво е повеќедимензионален процес кој комбинира различни елементи за да создаде извонредни, предизвикувачки и визуелно волшебни искуства во играта. Јасната структура, урамнотежената тежина, можностите за истражување, тематската естетика и итеративната префинетост се клучни за создавање нивоа што ги воодушевуваат играчите и ги одржуваат ангажирани во текот на нивното патување низ игрите.

Ангажирање на играчите

Ангажирањето на играчите, вклучува употреба на принципи и стратегии за дизајн за да се долови и одржи нивниот интерес во текот на искуството со игрите. Ова се постигнува со создавање на извонредно и интерактивно опкружување кое ја поттикнува љубопитноста, емоциите и мотивацијата на играчите, одржувајќи ги вложени во играта и желни да продолжат да играат. [4, 91]

Клучниот принцип на ангажирање на играчите е нудење значајна и привлечна игра. Игрите треба да обезбедат различни предизвици, цели и награди кои се грижат за различните преференци на играчите и нивоа на нивните вештини. Постигнувањето на вистинската рамнотежа помеѓу тежината и остварливиот напредок создава чувство на достигнување и задоволство, мотивирајќи ги играчите да истражуваат повеќе од играта. [25]

Негувањето на силна емоционална врска помеѓу играчите и светот на играта е исто така важно. Ова може да се постигне со развивање на релативни ликови, волшебни приказни и незаборавни поставки. Привлечните наративи, добро изработениот дијалог и влијателниот визуелен и аудио дизајн придонесуваат за извонредно искуство во кое играчите можат емотивно да се инвестираат. [120]

Промовирањето на социјалната интеракција и ангажманот во заедницата е моќна алатка за задржување и уживање на играчите. Карактеристиките со повеќе играчи, опциите за кооперативна игра обезбедуваат можности за играчите да се поврзат со другите, поттикнувајќи чувство на припадност и здрава конкуренција. Дополнително, интегрирањето на функционалностите за социјално споделување и креирањето и споделувањето содржини генерирани од корисниците ја подобруваат долговечноста на играта и вклученоста на заедницата. [73, 89]

Континуираното ажурирање на играта со нови содржини, функции и подобрувања е од клучно значење за одржување на ангажираноста на играчите. Редовните ажурирања ја одржуваат играта свежа и возбудлива, давајќи им на играчите причина повторно да ја посетат и да го продолжат своето виртуелно патување. Активното слушање на повратните информации од играчите и инкорпорирање на нивните предлози поттикнува чувство на сопственост на играчите и инвестирање во еволуцијата на играта.

Накратко---, ангажманот на играчите се врти околу создавање искуство што го привлекува вниманието на играчите, предизвикува емоции и обезбедува чувство на исполнетост. Со користење на внимателен дизајн на игри, значајно раскажување приказни, социјална интеракција и редовни ажурирања, програмерите можат да негуваат посветена и ентузијастичка база на играчи, обезбедувајќи долгорочен успех и уживање во нивната игра.

1.2 Психологија на играчите во игрите

Во овој дел, фокусот е на разбирање на психологијата на играчите и како тоа влијае на нивното однесување и искуството во игрите. Опфаќа теми како што се мотивација, типови на играчи, задоволство на играчите, доживување (immersion)[2] и влијанието на емоциите врз играта. Ги испитува психолошките теории и модели релевантни за дизајнот на играта и дискутира како развивачите на игри можат да го искористат ова разбирање за да создадат поприврлечни искуства.

Мотивација

Мотивацијата, влијае на ангажираноста и уживањето на играчите. За да се мотивираат играчите, треба да се обезбедат јасни и остварливи цели. Остварливите цели ја вклучуваат желбата на играчите за постигнување и напредок. Наградите во играта, и надворешните награди, како што се виртуелната валута или козметичките надградби, мора да бидат избалансирани за да се создаде мотивациска јамка. Социјалната интеракција и конкуренцијата ја исполнуваат потребата на играчите за заедница и признание. Карактеристиките на повеќе играчи и табелите на водачи поттикнуваат соработка и повисоки достигнувања. Ефективните механизми за повратни информации, како визуелните знаци или наративните елементи, го зајакнуваат чувството за компетентност кај играчите и обезбедуваат можности за раст. Разбирањето на овие принципи им овозможува на развивачите на игри да создадат волшебни искуства кои го одржуваат ангажманот на играчите. [13, 128, 133]

[2] Дефиниција на immersion е длабоко ментално вклучување во нешто. Буквалниот превод на immersion е потопување, а мое толкување е зборот доживување.



1- 3 Модел на мотивација на играч

Ова истражување ги истражува различните мотивации за однесувањето во игрите. Теоријата за размена на труд/рекреација сугерира дека поединците бараат пријатни задачи откако ќе ги завршат сложените. Зголемените тешкотии во претходните задачи ја зголемуваат мотивацијата во ризичните игри. Игрите можат да ги исполнат имплицитните потреби, како што е намалувањето на стресот. Измислените игри ги задоволуваат барањата на децата за компетентност, социјална интеракција и решавање на конфликти. Факторите како социјалната интеракција, конкуренцијата, предизвикот, фантазијата, пренасочувањето и возбудата ги привлекуваат играчите со задоволување на психолошките потреби што недостасуваат во секојдневниот живот. Гејмерите имаат различни цели и стилови, вклучувајќи релаксација, забава, социјализација, уживање во приказната, мајсторство, достигнување и истражување. [13, 128, 133]



1- 4 Модел на проток поврзан со нивоата на предизвикот и вештината.

Луѓето имаат три основни психолошки потреби за задоволство: автономија, компетентност и поврзаност. Автономијата вклучува контрола над своите постапки, компетентноста е поттик за препознавање и вештина, а поврзаноста е потребата за поврзување и интеракција со другите. Видео игрите имаат потенцијал да ги исполнат овие потреби. Игрите можат да ја прошират автономијата преку избор на играчи и повеќе правци до завршување. Поврзаноста се задоволува преку интеракција во игрите со повеќе играчи и заедниците. Раните игри фокусирани на предизвици за исполнување на потребата од компетентност. Истражувачите го класифицираа ангажманот во игри врз основа на интелектуалните предизвици, физичките активности, сензорната стимулација, социјалните интеракции, наративната вклученост и емоционалните врски. Разбирањето на мотивациите на играчите може да го информира дизајнот на играта и да создаде привлечни искуства. [13, 128, 133]

Овој дел ни дава преглед на улогата на мотивацијата во развојот на игрите, мотивациите зад однесувањето во игрите и исполнувањето на основните психолошки потреби во искуствата со игрите. Со разгледување на овие фактори, програмерите можат да дизајнираат игри што ги воодушевуваат играчите и го одржуваат нивниот ангажман.

Типови на играчи

Кога станува збор за развој на игри, разбирањето на типовите на играчи е од суштинско значење за создавање привлечни и извонредни искуства. Играчите имаат различни преференции, мотивации и стилови на игра кои можат да се категоризираат во различни типови. Според моделот на Бартл, постојат четири главни типови на играчи: постигнувачи, убијци, социјализатори и истражувачи. [13]

Постигнувачите се мотивирани од постигнување на цели, отклучување на достигнувања и стекнување на награди. Тие уживаат да напредуваат низ играта и да ги совладуваат предизвиците за да стекнат чувство на мајсторство. Истражувачите, од друга страна, се водени од љубопитноста и возбудата на откривањето. Тие наоѓаат радост во истражувањето на светот на играта, откривање тајни и доживување нови области или приказни. [13]

Социјализаторите добиваат уживање од интеракцијата и поврзувањето со другите играчи. Тие напредуваат во средини со повеќе играчи каде што можат да соработуваат, да се натпреваруваат или едноставно да се дружат со колеги гејмери. За нив е важно да градат заедници и да се вклучат во кооперативна игра. Убијците, сепак, се конкурентни играчи кои добиваат задоволство од надминувањето на другите играчи. Тие уживаат во борба против играчите, стратешко размислување и демонстрација на своите вештини за да доминираат над другите играчи. [13]



1- 5 Модел на мотивација според теоријата на Бартл (1996).

Дополнително, има раскажувачи кои се воодушевени од наративот на играта. Тие ги ценат добро развиените ликови и извонредното раскажување приказни. Раскажувачите им даваат приоритет на богатите наративи и значајни избори кои го обликуваат светот на играта и патувањето на нивниот лик. [13]

Вреди да се напомене дека играчите можат да покажат особини од повеќе типови играчи, а нивните преференции може да се менуваат со текот на времето или во зависност од играта што ја играат. Моделот на Бартл зема во предвид две димензии: интересот на играчот за самата игра или другите играчи и фокусот на дејствување или интеракцијата. Постигнувачите и истражувачите уживаат во самата игра, додека убијците и социјалистите преферираат да играат со други играчи.

Ји (2006) дополнително го проширил моделот на Бартл и идентификувал три клучни компоненти на мотивацијата на играчите. Првата компонента е достигнување, кое вклучува желба за напредување, анализа на виртуелниот свет и подготвеност за натпреварување. Втората компонента е мотивацијата за дружење, која ја опфаќа желбата за случајна интеракција, градење односи и тимска работа. Третата компонента е доживувањето, која се состои од откривање на светот на играта, вклучување во играње со улоги, прилагодување на ликовите и бегство од реалниот живот. [133]

Со разгледување на овие типови играчи и мотивации за време на развојот на играта, програмерите можат да создадат приспособени искуства кои се привлечни на различни преференци на играчите, што резултира со позадоволувачка игра и зголемен ангажман на играчите.

Задоволство на играчите

Задоволството на играчите е клучно во развојот на играта бидејќи директно влијае на севкупниот успех на играта. За да се обезбеди задоволство на играчите, програмерите мора да ги разберат и да се грижат за психолошките потреби на нивната целна публика. Ова може да се постигне со обезбедување на ангажирана игра, значајни предизвици и наградни искуства кои создаваат чувство на достигнување и исполнетост за играчите. [25, 69, 92, 131]

Еден клучен принцип за задоволство на играчите е да се понуди чувство на прогресија. Играчите уживаат во чувството на раст и напредок во игра. Ова може да се постигне преку добро дизајнирани системи за израмнување, отклучување на нови способности или опрема и достигнување значајни пресвртници. Со обезбедување опипливи награди и јасно чувство за напредок, програмерите можат да ги задржат играчите мотивирани и задоволни. [25, 43, 92, 131]

Смисленото одлучување е уште еден важен аспект што го зголемува задоволството на играчите. Играчите ценат да имаат агенција и контрола врз нивните искуства во играта. Кога изборот влијае на приказната, развојот на ликот или резултатите од играта, играчите чувствуваат чувство на сопственост и доживување, што резултира со зголемено задоволство. [25, 69]

Урамнотежените нивоа на тежина се клучни за задоволството на играчите. Додека играчите треба да бидат доволно предизвикани за да почувствуваат чувство на достигнување, тежината не треба да доведе до фрустрација. Пронаоѓањето на вистинската рамнотежа обезбедува играчите да останат ангажирани и мотивирани да продолжат да играат, што на крајот води до зголемено задоволство. [25, 69, 92]

Социјалните интеракции и карактеристиките на повеќе играчи значително придонесуваат за задоволството на играчите. Многу играчи уживаат да се поврзуваат и да се натпреваруваат со други, без разлика дали преку кооперативно играње или онлајн режим за повеќе играчи. Со инкорпорирање на социјални елементи, игрите поттикнуваат чувство на заедница, поттикнуваат тимска работа и обезбедуваат дополнителен слој на уживање и задоволство. [25, 69, 92]

Вниманието на деталите и квалитетот се најважни за задоволството на играчите. Исполираната графика, мазната механика на игра, добро дизајнираните нивоа и искуствата без грешки придонесуваат за позитивно искуство на играчот. Со инвестирање во севкупниот квалитет на играта и навремено решавање на какви било технички проблеми, програмерите можат да се погрижат играчите да имаат непречено и извонредно искуство, што на крајот ќе го зголеми нивното задоволство. [25, 43, 69, 92]

Накратко, задоволството на играчите во развојот на играта бара разбирање на психологијата на играчите и грижа за нивните потреби и желби. Со фокусирање на

чувството за прогресија, значајно донесување одлуки, избалансирана тежина, социјални интеракции и квалитет, програмерите можат да создадат игри што ги пленат играчите и ги оставаат да се чувствуваат задоволни и исполнети. На крајот на краиштата, задоволството на играчите го поттикнува ангажирањето на играчите и може да доведе до долгорочен успех на играта.

Доживување (Immersion)

Доживувањето во видео- игрите има за цел да ги плени играчите и да ги пренесе во виртуелен свет. Тоа оди подалеку од едноставно уживање во елементите на играта како графика и приказна; тоа е психолошки феномен кој се јавува кога играчите длабоко се апсорбираат во богатиот и детален свет на играта. [54, 68, 77, 87, 112]

За да се постигне доживувањето, развивачите на игри користат различни принципи и техники. Тие создаваат конзистентен и кохезивен свет на игри со логични правила и однесувања, дозволувајќи им на играчите да го прекинат своето неверување и да ја прифатат виртуелната реалност што им е претставена. Агенцијата на играчите, или способноста да се прават значајни избори и да се има контрола врз акциите, го подобрува доживувањето со тоа што ги прави играчите да се чувствуваат поврзани и ангажирани. [54, 112, 125]

Интегрираната теорија на мотивација		
	Интерес во светот	Интерес во други играчи
Постигнување на цел	Достигнување, Мајстор на играта, Фантазија: стекнување на моќ и вештини	Победа на натпреварот
Интеракција	Доживување, Откривање на светот, Играње улоги, Пренасочување: бегство од реалниот живот	Припадност, Дружење, Тимска работа

Вниманието на деталите е исто така од суштинско значење за доживување. Прецизно изработените елементи како што се реалистичниот дизајн на звук, визуелните ефекти и градењето на светот придонесуваат за мултисензорно искуство што ги вовлекува играчите подлабоко во универзумот на играта. Дополнително, наративната структура и темпото на играта играат значајна улога во одржувањето на доживувањето. Привлечната приказна со добро развиени ликови и пресврти на заплетот емотивно ги поврзува играчите со светот на играта, додека соодветното темпо ги одржува ангажирани. [54, 77, 112, 125]

Разбирањето на психологијата на играчите е од клучно значење за создавање на извонредни искуства. Програмерите на игри ги земаат во предвид мотивациите, желбите и очекувањата на играчите, привлекувајќи ги нивните внатрешни потреби за љубопитност, достигнувања и социјална интеракција. Со усогласување на играта со овие психолошки аспекти, програмерите можат да ги држат играчите ангажирани подолго време. [25, 87]

Накратко, доживувањето во играта е за создавање беспрекорно и волшебно искуство кое им овозможува на играчите целосно да се вклучат во виртуелниот свет. Доследноста, агенцијата на играчот, вниманието на деталите, наративната структура и разбирањето на психологијата на играчите се сите клучни елементи кои придонесуваат за градење извонредни искуства. Со примена на овие принципи, развивачите на игри можат да ги пренесат играчите во извонредни виртуелни области и да им овозможат незаборавни гејмерски искуства.

Влијанието од емоциите врз играта

Разбирањето на влијанието на емоциите врз играта е фасцинантен аспект што треба да го разгледаат развивачите на игри. Емоциите играат клучна улога во обликувањето на искуствата и ангажираноста на играчите со светот на играта. Позитивните емоции како возбуда, радост и љубопитност во голема мера ја подобруваат играта со поттикнување на истражување, преземање ризик и надминување на предизвиците. Тие обезбедуваат чувство на достигнување и награда кога играчите успеваат, како и поттикнуваат врска помеѓу играчите и ликовите или

приказната на играта, зголемувајќи ја нивната веројатност за континуирана игра и препорака до други луѓе. [25, 49, 58, 84, 96]

Од друга страна, негативните емоции како што се: стравот, фрустрацијата и лутината исто така можат значително да влијаат на играта. Додека прекумерната фрустрација или гнев може да доведе до незадоволство или напуштање на играчите, добро поставените моменти на напнатост или страв можат да создадат извонредни искуства. Стравот ги зголемува сетилата и емоционалната инвестиција на играчите, што го прави успехот во играта поисплатлив. Надминувањето на првично фрустрирачките предизвици, исто така, може да обезбеди чувство на достигнување и да ги мотивира играчите да истраат. [45, 49, 96, 110]

За да предизвикаат специфични емоции, развивачите на игри стратешки дизајнираат искуства за играње користејќи наративни, аудиовизуелни знаци, механика и дизајн на ниво. Со создавање на емотивни лакови кои резонираат со играчите, програмерите имаат за цел да го подобрат уживањето и доживувањето, оставајќи трајно влијание врз сеќавањата на играчите и желбата да се вратат во играта. [49, 94]

Истражувачите спроведоа студии кои ги анализираа емоциите за време на играта со комбинирање на анализа на самоизвестување, неврофизиолошка активност и изрази на однесување. Овие студии ги испитуваат различните емоционални компоненти и физиолошките одговори на одредени настани на играта. Тие идентификувале специфични емоционални одговори, како што се фрустрација, страв, изненадување, љубопитност, одвратност, релаксација, апсорпција и чувство на достигнување, врз основа на самоизвештаи и набљудувања на емоционални знаци за време на играта. Дополнително, истражувачите проценуваа дискретни емоции користејќи методи за самоизвестување и физиолошки одговори, наоѓајќи асоцијации помеѓу специфични емоции и настани од играта, како и нивното влијание врз физиолошките одговори и субјективните искуства на играчите. [49]

Понатаму, истражувачите ја истражуваа врската помеѓу физиолошките одговори, нервната активност и емоционалните искуства за време на играта. Промените во физиолошките реакции, како што се отчукувањата на срцето и спроводливоста на кожата, се случуваат во одредени периоди во текот на играта,

особено во ситуации на непосредна опасност за ликот на играта. Невралните одговори, анализирани преку методи како електроенцефалографија (ERP) и функционална магнетна резонанца (fMRI), обезбедуваат увид во моделите на кортикална активност поврзани со позитивни и негативни емоции. [49]

Со разгледување на димензионалните и категоричните модели на емоции, истражувачите развија методи за континуирано моделирање на емоциите користејќи физиолошки податоци. Овие методи ги пресметуваат вредностите на возбуда и валентни од физиолошките сигнали и ги користат за одредување конкретни емоции, како што се досада, предизвик, возбуда, фрустрација и забава. Анализирањето на физиолошките индекси на емоциите на играчите нуди вредни сознанија за искуствата на корисниците и може да се примени и за други форми на забава. [49]

Накратко, влијанието на емоциите врз играта е сложен феномен што развивачите на игри и истражувачите го признаваат. Тие ја препознаваат важноста од разбирање на психологијата на играчите и дизајнирање игри кои стратешки предизвикуваат специфични емоции. Со внимателно балансирање на позитивните и негативните емоции, програмерите можат да создадат привлечни, незаборавни и значајни искуства за играчите. Интегрирањето на анализата за самоизвестување, неврофизиолошката активност и изразите на однесување во истражувањето обезбедува сеопфатно разбирање на емоционалните одговори за време на играта, придонесувајќи за подобрување на ангажираноста на играчите, задоволството и севкупниот успех на видео игрите.

1.3 Техники на раскажување на приказните во игрите

Оваа потсекција ги истражува техниките и стратегиите што се користат за создавање привлечни наративи во игрите. Опфаќа теми како наративни структури, развој на ликови, градење свет и интерактивно раскажување приказни. Се дискутира за различните пристапи на раскажување приказни во игрите и како наративните елементи можат да го подобрат ангажирањето и доживувањето на играчите.

Наративна структура

Наративната структура го обликува целокупното искуство на раскажување приказни за играчите. Структурата од три чина најчесто се користи во игрите, опфаќајќи го поставувањето, конфронтацијата и решавањето. [9, 61]

Во првиот чин, поставувањето, играчите се запознаваат со светот на играта, ликовите и нивните мотиви. Се утврдува првичниот конфликт или цел што го поттикнува наративот напред, обезбедувајќи информации во позадината за играчите да ја доживуваат приказната. [9, 61]

Конфронтацијата, или вториот чин, носи предизвици и конфликти кои ги тестираат вештините на играчот и ја унапредуваат приказната. Овој чин се карактеризира со зголемена акција, клучни моменти и зголемена напнатост, создавајќи возбуда. [9, 61]

Решавањето, или третиот чин, ја достигнува кулминацијата на наративот и обезбедува крај. Ги вклучува постапките и одлуките на играчот, што доведува до конечна конфронтација или предизвик што мора да се надмине. Овој чин им овозможува на играчите да доживеат задоволство додека наративните нишки се спојуваат и доаѓаат до заклучок. [9, 61]

Иако структурата со три чина е вообичаена, развивачите на игри имаат слобода да експериментираат со алтернативни структури засновани на жанрот, темите и механиката на играта. Некои игри користат нелинеарни структури, разгранети патеки или повеќе перспективи, дозволувајќи им на играчите да ја обликуваат приказната преку нивниот избор. [9, 32, 61]

Методот низа бисери претставува неинтерактивна приказна преку текст, сцени или други формати, прошарани со игра. Овој метод нуди контрола врз темпото и другите аспекти на приказната, но играчите може да чувствуваат недостаток на агенција. [32]

Од друга страна, методот на машина за приказни вклучува елементи на играта кои дејствуваат како генератор на приказни. Играчите создаваат свои приказни преку интеракции со светот на играта, обезбедувајќи мноштво можности. Сепак, програмерите имаат помала контрола врз приказните што се појавуваат. [32]

Наративот за разгранување комбинира елементи од методот низа бисери со избор на играчите. Одлуките на играчот одредуваат кои пропишани сцени од приказната се прикажуваат. Тој нуди позначаен избор, но може да бара ресурси и може да доведе до фаворизирани патеки. [32]

Структурата на забавниот парк комбинира контролирани приказни со нова игра. Играчите можат да изберат различни наративни патеки во голем свет, обезбедувајќи разновидни и интересни искуства. Сепак, поддршката на повеќе наративи во таков амбиент може да биде скапа. [32]

За крај, техниката лов на чистачи вклучува ставање елементи на приказната во средината на играта за играчите да ги откријат со свое темпо. Овие опционални фрагменти може да се најдат по кој било редослед, дозволувајќи им на играчите да ја истражуваат и откријат приказната на нелинеарен начин. [32]

Со избирање и прилагодување на овие наративни структури, развивачите на игри можат да создадат привлечни искуства за раскажување приказни кои ја надополнуваат механиката на играта и доживувањето на играчите во светот на играта.

Развој на карактерот

Развојот на карактерите, внесува длабочина и доживување во искуствата на играчите. Тоа вклучува изработка на релативни ликови кои се развиваат и растат во текот на играта. Ефективниот развој вклучува богата историја, посебни особини на личноста и значајни односи со другите ликови и светот на играта. [8, 37]

Добро развиениот лик има позадина што обезбедува контекст и мотивација за нивните постапки, создавајќи емпатија и разбирање. Уникатните особини на личноста ги прават ликовите незаборавни и тродимензионални, рефлектирани во нивните дијалози и интеракции. Смислените односи со другите ликови ги предизвикуваат верувањата и обезбедуваат можности за раст. [51, 75, 106]

Во игрите, ликовите мора да избегнуваат да останат стагнантни, пркосејќи им на природните закони на раскажување приказни. Ситните знаци можеби нема да бараат промена, но главните ликови, и PC и NPC, мора да претрпат раст и развој. Растот ги опишува промените додека ликовите напредуваат, додека развојот ја открива нивната вистинска природа. [36]

Развојот на карактерот ги плени играчите, предизвикува емоции и создава извонредни искуства. Со изработка на добро заоблени ликови, развивачите на игри обезбедуваат привлечни наративи кои резонираат со играчите. [91, 100]

Градење на светот

Градењето на светот во развојот на игри е клучен процес за создавање на извонредни виртуелни средини за играчите. Тоа вклучува изградба на детално измислени светови со богата историја, различни култури и волшебни пејзажи. Програмерите црпат инспирација од различни извори за да ги интегрираат влијанијата во наративот и околината на играта, стремејќи се кон автентичност и длабочина. [10, 53, 124]

Вниманието на деталите е од суштинско значење, бидејќи секој аспект на светот мора внимателно да се разгледа, од архитектура до општествени структури. Успешното градење на светот вклучува техники на раскажување приказни кои го

подобруваат доживувањето на играчите и емоционалното вложување. Ликовите, главни или споредни, имаат добро дефинирани личности и мотиви, кои ги поттикнуваат нивните постапки. Ангажираните наративи и значајните потраги им даваат на играчите чувство за цел и напредок. [10, 53, 124]

Светот треба да се чувствува жив и одговорен, со динамични екосистеми, променливо време и интерактивни елементи. Ликовите кои не се играчи со рутини и однесувања го зголемуваат реализмот, создавајќи можности за нова игра. Механиката за играње ја надополнува светската тема и наратив, подобрувајќи го доживувањето и поврзувањето со играчите. [10, 53, 124]

Градењето на светот ги поставува правилата и ограничувањата во кои се одвива приказната. Ги одредува јазиците, општествените структури, валутите и многу повеќе. Тоа е основата што им дава значење и контекст на приказната и дејствијата во светот на играта. [10, 53, 124]

Различни дисциплини работат заедно во градењето на светот, при што директорот на играта обезбедува кохезивен свет. Дизајнерите на нивоа и уметниците од околината соработуваат за да создадат функционални простори што ги задоволуваат потребите за игра и приказна [10, 53, 124]

Градењето на светот може да започне во претпродукција и да продолжи во текот на развојот. Важно е да постигнете рамнотежа и да не се преоптоварувате. Фокусот треба да биде на раскажувањето на приказната со светот, усогласување со тонот и темата на играта. [10, 53, 124]

Интерактивно раскажување на приказните

Интерактивното раскажување приказни е волшебен пристап кон развојот на играта што го става играчот во центарот на наративното искуство. Таа има за цел да ги ангажира играчите на подлабоко ниво дозволувајќи им да прават избори и да влијаат на исходот на приказната. Во интерактивното раскажување приказни, играчите стануваат активни учесници, обликувајќи ја насоката и развојот на наративот преку нивните одлуки и постапки.

Еден од основните принципи на интерактивното раскажување приказни е обезбедувањето значајни избори. Играчите треба да чувствуваат дека нивните одлуки имаат значително влијание врз приказната и нејзините ликови. Овие избори може да се движат од морални дилеми кои ги тестираат вредностите на играчот до стратешки одлуки кои влијаат на исходот на заплетот. Со нудење значајни избори, програмерите создаваат чувство на агенција, правејќи ги играчите да се чувствуваат лично инвестирани во приказната и мотивирани да истражуваат различни патишта и резултати. [20]

Друг важен аспект на интерактивното раскажување е концептот на последица и разгранување наративи. Дејствата и изборите направени од играчите треба да имаат последици кои одекнуваат низ приказната, обликувајќи ги следните настани и менувајќи ги односите на карактерите. Оваа динамична природа на раскажување приказни осигурува дека секое играње може да биде единствено, нудејќи им на играчите чувство на откривање и вредност за повторување. [101]

Понатаму, доживувањето, игра витална улога во интерактивното раскажување приказни. Програмерите се стремат да создадат беспрекорна мешавина помеѓу механиката на игра и наративните елементи, овозможувајќи им на играчите целосно да се нурнат во светот на играта. Со користење на техники како што се привлечен дијалог, атмосферски средини и реални интеракции со карактери, програмерите можат да поттикнат емоционални врски меѓу играчите и приказната, зголемувајќи ја нивната инвестиција и ангажман. [55]

Покрај тоа, успешното интерактивно раскажување често вклучува елементи на истражување и откривање. Светот на играта треба да биде богат со детали, скриени тајни и изборни приказни што играчите можат да ги откријат со активно истражување на нивната околина. Ова ги охрабрува играчите да се вклучат во наративот надвор од главната линија, наградувајќи ја нивната љубопитност и обезбедувајќи посеопфатно и потопливо искуство. [2]

Првата голема дебата за раскажување игри беше фокусирана на прашањето дали на игрите воопшто им се потребни приказни за почеток. Сепак, таа дебата во голема мера е завршена, а одговорот сега се разбира дека е повеќе нијансиран. Некои игри, како што се спортските и игрите со загатки, првенствено се фокусираат на

нивната основна игра и можеби нема да бараат формална приказна. Од друга страна, авантуристичките игри и игрите со улоги често привлекуваат обожаватели кои ги ценат интересните ликови и длабоките приказни. Иако на игрите не им се „потребни“ приказни, добрата приказна може да ја подобри речиси секоја игра и да го подобри целокупното искуство.

Како што еволуираше индустријата за игри, растеше признавањето на важноста на приказните во игрите. Жанровите како стрелците во прво лице и акционите игри, кои некогаш беа познати по основните или без приказни, почнаа да вклучуваат посложени наративи. Фокусот се префрли на создавање игри фокусирани на приказната кои на играчите им нудат повеќе избор и контрола врз наративот. Многумина во индустријата гледаат на високо интерактивни приказни водени од играчи како иднина на раскажувањето приказни за игри, додека други тврдат дека потрадиционалните форми на раскажување сè уште имаат вредност.

Дефинирањето на интерактивно раскажување приказни може да биде предизвик поради спротивставените дефиниции и толкувања. Додека некои ја сметаат секоја форма на интеракција со приказна, како што е вртење страница или притискање на копче, како интерактивно раскажување, вистинската интерактивност подразбира дека играчите можат директно да се вклучат и да влијаат на ликовите или поставката на приказната. Читањето фусноти или интеракцијата со физичките аспекти на книгата, како во „Куќа од лисја“, не се квалификува како интерактивно раскажување. Сепак, не е неопходно играчите да имаат целосна контрола врз приказната за таа да се смета за интерактивна; едноставно им е потребна способност на некој начин да комуницираат со наративот.

Како заклучок, интерактивното раскажување ги овластува играчите да ги обликуваат сопствените наративни искуства во игрите. Тоа вклучува значајни избори, последици и разгранети наративи, доживување и елементи на истражување и откривање. Иако повеќе не се дебатира за потребата од приказни во игрите, дискусиите сега се вртат околу тоа кои видови приказни се најдобри и колкава контрола треба да имаат играчите врз наративот. Интерактивното раскажување отвара нови можности за привлечни и извонредни искуства со играта, замаглувајќи ги линиите помеѓу играта и наративот и дозволувајќи им на играчите да станат активни учесници во приказните што ги играат.

1.4 Анализа на механиката на игрите

Оваа потсекција се фокусира на анализа на механиката на играта и нивното влијание врз играта и искуството на играчите. Опфаќа теми како што се балансирање на игри, системи за наградување, механика на прогресија и интеракција помеѓу механиката и однесувањето на играчите. Испитува како механиката на играта може да влијае на одлуките на играчите, развојот на вештините и целокупното уживање во играта.

Балансирање на играта

Балансот на играта има за цел да создаде фер и пријатно искуство за играчите. Тоа вклучува внимателно прилагодување на различни елементи во играта за да се осигура дека ниту една стратегија или играч нема огромна предност во однос на другите. Постигнувањето на рамнотежа на играта бара прецизно разгледување на факторите како што се способностите на карактерите, јаките точки, нивоата на тежина и целокупната механика на играта. [3, 83, 85, 108, 111, 119]

Еден клучен принцип на рамнотежата на играта е обезбедување значаен избор на играчите. Ова значи дека различните опции или стратегии треба да имаат споредливи јаки и слаби страни, дозволувајќи им на играчите да донесуваат тактички одлуки врз основа на нивните преференции и стилови на игра. Нудејќи разновиден опсег на остварливи избори, рамнотежата на играта промовира стратешко размислување и ги охрабрува играчите да истражуваат различни пристапи за надминување на предизвиците. [3, 85, 108]

Друг важен аспект на рамнотежата на играта е кривата на прогресија. Добро избалансирана игра обезбедува чувство на прогресија и раст, дозволувајќи им на играчите да почувствуваат чувство на достигнување додека напредуваат низ играта. Ова подразбира внимателно калибрирање на тежината на предизвиците за да се совпадне со зголеменото ниво на вештини на играчот и нивото на моќност на нивните ликови или способности. Постепеното и урамнотежено зголемување на тежината гарантира дека играчите се постојано ангажирани и мотивирани да ги надминат пречките. [3, 83, 85, 111]

Покрај тоа, рамнотежата на играта е тесно поврзана со концептот на ризик и награда. Тоа вклучува балансирање на потенцијалните награди на одредени активности или стратегии со поврзаните ризици. Дејствата со повисок ризик треба да понудат поголеми награди, додека опциите со помал ризик обезбедуваат поконзистентни, но умерени придобивки. Оваа рамнотежа на ризик и награда ги охрабрува играчите да преземаат пресметани ризици, поттикнувајќи чувство на возбуда и постигнување кога нивните одлуки се исплатат. [3, 85, 108, 119]

За крај, рамнотежата на играта е тековен процес кој бара континуирано следење и прилагодувања. Повратните информации од играчите и анализата на податоците играат витална улога во идентификувањето на областите кои може да бидат неурамнотежени или кои имаат потреба од подобрување. Програмерите на игри мора да реагираат на повратните информации на заедницата и да бидат подготвени да ги направат неопходните промени за да одржат пријатно и фер-искуство со игрите. [3, 83, 111]

Накратко, рамнотежата на играта е основен принцип во развојот на играта кој има за цел да создаде фер и пријатно искуство за играчите. Со обезбедување значајни избори, управување со прогресијата, балансирање на ризикот и наградата и континуирано следење на рамнотежата на играта, програмерите можат да создадат привлечна и извонредна игра што ги тера играчите да се вратат за повеќе.

Системи за наградување

Системите за наградување во механиката на игрите играат клучна улога во мотивирањето и поттикнувањето на играчите да се вклучат во играта и да напредуваат преку нејзината содржина. Овие системи се дизајнирани да им овозможат на играчите чувство на достигнување, задоволство и напредок преку нудење награди за нивните постапки и достигнувања. Наградите може да имаат различни форми, вклучувајќи валута во играта, ставки, поени за искуство, козметички надградби или отклучување нови нивоа или содржини. [46, 93, 106]

Една од клучните функции на системите за наградување е да го зајакне саканото однесување на играчите. Со обезбедување награди за специфични акции

или пресвртници, дизајнерите на игри можат да ги охрабрат играчите да се вклучат во активности кои се усогласени со целите на играта или наменетиот стил на игра. На пример, игра која го нагласува истражувањето може да ги награди играчите за откривање скриени области или собирање ретки артефакти, додека игра фокусирана на борба може да понуди награди за поразување на предизвикувачки непријатели или завршување на тешки мисии. Овие награди служат како позитивно засилување, сигнализирајќи им на играчите дека напредуваат и даваат значаен придонес за нивното патување во играта. [27, 70, 104, 113]

Добро дизајнираните системи за наградување прават рамнотежа помеѓу краткорочното задоволство и долгорочните цели. Краткорочните награди нудат непосредна повратна информација и задоволство, обезбедувајќи моментално чувство на постигнување. Ова може да вклучува мали награди за завршување помали задачи, надминување помали предизвици или постигнување пресвртници во рамките на сесијата на играта. Од друга страна, долгорочните награди се насочени кон поголеми достигнувања или значителен напредок во подолги периоди. Овие награди се често позначајни и може да вклучуваат отклучување нови способности, пристап до напредна опрема или достигнување важни пресвртници во приказната. Комбинацијата на краткорочни и долгорочни награди создава чувство на исчекување, прогресија и чувство на вложување во играта. [27, 47, 93, 104, 113]

Покрај мотивирањето на акциите на играчите, системите за наградување можат да го подобрат и ангажирањето и задржувањето на играчите. Со стратешко дизајнирање на награди кои се усогласени со преференциите и желбите на играчите, програмерите можат да создадат чувство на возбуда и исчекување. Ова може да се постигне преку различни методи, како што се воведување случајни или ретки награди кои обезбедуваат чувство на изненадување и возбуда, имплементација на системи за достигнувања кои препознаваат и наградуваат конкретни достигнувања или инкорпорирање на социјални награди кои им овозможуваат на играчите да ги покажат своите достигнувања на другите. [46, 93, 113]

Сепак, важно е дизајнерите на игри да се погрижат системите за наградување да бидат добро избалансирани и да не станат премногу експлоатирачки или нарушувачки за играта. Ако наградите премногу лесно се добиваат или немаат значајно влијание, тие може да го намалат чувството за предизвик и да ја поткопаат

долговечноста на играта. Спротивно на тоа, ако наградите се премногу тешки за постигнување или се чувствуваат недостижни, играчите може да станат фрустрирани или незаинтересирани. Постигнувањето на вистинската рамнотежа бара внимателно разгледување на темпото на играта, вредноста и влијанието на наградите и одржување на чувството за правичност и прогресија во текот на патувањето на играчот. [27, 46, 70, 113]

Како заклучок, системите за наградување се моќна компонента на механиката на играта која ја поттикнува мотивацијата, ангажираноста и прогресијата на играчите. Со нудење награди за посакуваните акции и достигнувања на играчите, програмерите можат да создадат чувство на достигнување, задоволство и вложување во играта. Добро дизајнираните системи за наградување постигнуваат рамнотежа помеѓу краткорочното задоволување и долгорочните цели, обезбедувајќи играчите постојано да се мотивирани да истражуваат, да ги надминат предизвиците и да го продолжат своето патување во светот на играта.

Механики на прогресија

Механиката на прогресијата е основен аспект на дизајнот на играта која има за цел да им обезбеди на играчите чувство на раст, развој и напредок во текот на нивното искуство во игра. Овие механики се дизајнирани да создадат чувство на прогресија со постепено отклучување на нови содржини, способности, предизвици или наративни елементи додека играчите вложуваат време и напор во играта. [91, 108, 114]

Една од основните цели на механиката за прогресија е да обезбеди чувство на достигнување и награда за акциите на играчите. Ова може да се постигне преку различни методи, како што се израмнување на карактерот, отклучување на нови вештини или способности, стекнување моќна опрема или пристап до претходно недостапни области. Како што играчите напредуваат, тие чувствуваат чувство на мајсторство над механиката на играта и стануваат поспособни да ги надминат предизвиците, што пак ја поттикнува нивната мотивација да продолжат да играат. [25, 56, 106]

Механиката за прогресија често користи структуриран систем на поени за искуство (XP) или други форми на валута за следење и претставување на напредокот на играчот. Со акумулирање на искуство или валута преку активности за играње, играчите можат да заработат награди и да отклучат нови можности. Овој систем им овозможува на играчите да видат опипливи докази за нивниот напредок и дава јасен показател за нивното моментално ниво на вештина или статус во играта. [56, 91, 105]

Ефективната механика за прогресија е дизајнирана да ги балансира темпото и тешкотиите, осигурувајќи дека играчите се постојано ангажирани и предизвикувани. Постепеното воведување на нови содржини, предизвици или способности како што напредуваат играчите помага да се одржи чувството на новина и спречува играта да стане повторлива или да стагнира. Во исто време, тежината на играта треба да се зголемува во тандем со прогресијата на играчот за да се обезбеди соодветно ниво на предизвик и да се избегнат поразителни или фрустрирачки емоции. [25, 108, 114]

Добро дизајнираниот систем за прогресија, исто така, им нуди на играчите значајни избори и опции за прилагодување. Ова може да вклучува стебла на вештини или системи за таленти кои им овозможуваат на играчите да се специјализираат и да ги приспособат способностите на нивниот карактер за да одговараат на нивниот стил на игра, или разгранување на наративни патеки кои нудат различни резултати врз основа на одлуките на играчите. Способноста да се обликува развојот на ликот или приказната на играта според преференциите на играчот, ја подобрува агенцијата на играчот и инвестирањето во играта. [56, 91, 105]

Социјалните и кооперативните аспекти, исто така, може да се интегрираат во механиката на прогресија, дозволувајќи им на играчите да соработуваат или да се натпреваруваат со други. Ова може да биде во форма на системи за рангирање со повеќе играчи, кооперативни предизвици или табли на водачи кои ги поттикнуваат играчите да се стремат кон повисоки рангирања или да постигнат конкретни цели. Со инкорпорирање на социјални елементи, механиката на прогресија може да поттикне чувство на заедница, здрава конкуренција и споделени достигнувања меѓу играчите. [56, 108, 114]

Важно е дизајнерите на игри да го земат во предвид темпото и рамнотежата на механиката на прогресија за да се осигураат дека играчите чувствуваат постојано

чувство на раст и достигнување. Напредокот треба да биде доволно предизвикувачки за да обезбеди чувство на достигнување, но не толку тешко што ќе стане пречка за напредокот на играчите. Балансирањето на стапката на прогресија и наградите поврзани со него помага да се одржи ангажираноста и задоволството на играчите во текот на играта. [25, 105, 108, 114]

Накратко, механиката на прогресија е од суштинско значење во дизајнот на играта бидејќи им обезбедува на играчите чувство на раст, достигнување и напредување. Со постепено отклучување на нови содржини, предизвици, способности или наративни елементи, овие механики создаваат привлечно искуство во игра што ги мотивира играчите да продолжат да играат и да инвестираат време и напор во играта. Добро избалансиран и забрзан систем за прогресија гарантира дека играчите постојано се предизвикуваат, наградуваат и се ангажираат во текот на нивното патување во светот на играта.

Интеракција помеѓу механиката и однесувањето на играчите

Интеракцијата помеѓу механиката на играта и однесувањето на играчите е критичен аспект на дизајнот на играта што го обликува целокупното искуство на играчот. Механиката на играта влијае на тоа како играчите комуницираат со светот на играта, додека однесувањето на играчите, пак, може да биде под влијание и обликувано од механиката со која се среќаваат. Оваа динамична врска помеѓу механиката и однесувањето на играчите создава јамка за повратни информации што влијае на играта и целокупното уживање во играта. [3, 65, 91, 108, 118]

Механиката на играта може да го обликува однесувањето на играчите преку обезбедување стимулации и последици за конкретни дејства. Добро дизајнираната механика создава рамка која ги поттикнува играчите да се вклучат во посакуваното однесување и ги обесхрабрува несаканите. На пример, игра која наградува кооперативна игра и тимска работа може да ги поттикне играчите да соработуваат и ефективно да комуницираат за да ги надминат предизвиците. Спротивно на тоа, механиката која казнува или обесхрабрува одредени дејствија, како што е губењето на здравјето или ресурсите за непромислено однесување, може да ги поттикне играчите да прифатат постратешки и промислени пристапи. [1, 26, 30, 39, 44, 73, 130]

Однесувањето на играчите, од друга страна, исто така може да влијае на ефективноста и влијанието на механиката на играта. Играчите можат да истражуваат, експериментираат и да најдат креативни начини за интеракција со механиката, понекогаш дури и откривајќи несакани стратегии или пристапи. Ова однесување управувано од играчите може да доведе до појава на игра, каде што се случуваат неочекувани интеракции и исходи, додавајќи длабочина и богатство на искуството на играта. Дизајнерите на игри можат да набљудуваат и да учат од однесувањето на играчите за да ја усовршат и повторуваат нивната механика, дополнително подобрувајќи ја играта. [1, 5, 24, 26, 91, 106, 107, 130]

Интеракцијата помеѓу механиката и однесувањето на играчите може да се види и во начинот на кој играчите се прилагодуваат и учат со текот на времето. Како што играчите стекнуваат искуство со игра, тие развиваат стратегии и техники за оптимизирање на нивните перформанси. Тие можат да откријат кратенки, да ги искористат слабостите или да откријат скриени тајни во механиката. Како одговор, дизајнерите на игри можат да воведат нова механика или да ги менуваат постојните за да одржат рамнотежа и да ги предизвикаат играчите да го прилагодат своето однесување. Овој итеративен процес помага да се обликува искуството на играчите и осигурува дека играта ќе остане привлечна и наградувачка. [5, 24, 65, 73, 106, 107, 108, 130]

Важно е дизајнерите на игри да го земат во предвид потенцијалното влијание на механиката врз однесувањето на играчите и обратно. Тие треба внимателно да анализираат како механиката може да ги обликува акциите на играчите и како играчите би можеле да ги интерпретираат и да одговорат на тие механики. Со разбирање на моделите и мотивациите на однесувањето на играчите, дизајнерите можат да создадат механика што се усогласува со очекувањата, преференциите и стилот на игра на играчите, поттикнувајќи чувство на агенција и доживување. Редовното тестирање за играње и собирање повратни информации од играчите може да обезбеди вредни сознанија за интеракцијата помеѓу механиката и однесувањето на играчите, дозволувајќи им на дизајнерите да ги усовршат и подобрат своите одлуки за дизајнирање на играта. [1, 3, 26, 44, 91, 106, 107, 118]

Како заклучок, интеракцијата помеѓу механиката на играта и однесувањето на играчите е динамична и реципрочна врска што го обликува искуството во играта.

Механиката влијае на однесувањето на играчите преку обезбедување на стимулации и последици, додека однесувањето на играчот, пак, може да влијае на ефективноста и толкувањето на механиката. Со разбирање на оваа интеракција, дизајнерите на игри можат да создадат привлечни и извонредни искуства кои резонираат со играчите и обезбедуваат значајни можности за интеракција и игра.

1.5 Шаблони на дизајн во развојот на игри

Овде, фокусот е на истражување на заедничките модели на дизајн и најдобрите практики што се користат во развојот на игри. Опфаќа теми како што се објектно-ориентиран дизајн, дизајн базиран на компоненти, дизајн управуван од настани и други модели на дизајн специфични за развојот на игри. Оваа потсекција дискутира како овие модели на дизајн може да ја подобрат организацијата на кодот, модуларноста и одржливоста во проектите за развој на игри.

Објектно-ориентиран дизајн

Објектно-ориентираниот дизајн (OOD) е фундаментален принцип во развојот на играта што се врти околу организирање на кодот и податоците во повеќекратно употребливи и модуларни објекти. Промовира структуриран пристап кон дизајнот на софтвер со инкапсулирање на сродни својства и однесувања во кохезивни единици наречени објекти. Овие објекти комуницираат едни со други преку добро дефинирани интерфејси, овозможувајќи полесно одржување, проширување и повторна употреба на кодот. [6, 38, 60, 80]

Во објектно-ориентираниот дизајн, класите служат како нацрти за создавање објекти. Класата ги дефинира атрибутите (податоци) и методите (однесувањето) што ќе ги поседуваат објектите од таа класа. Енкапсулацијата е клучен концепт во OOD, бидејќи вклучува здружување на податоци и методи заедно во рамките на класата и изложување само на потребните интерфејси за интеракција со објектот. Оваа инкапсулација помага во постигнување апстракција на податоците и криење на внатрешните детали за имплементацијата, промовирање на модуларноста на кодот и намалување на зависностите. [15, 31, 64, 80]

Наследувањето е уште еден важен принцип во OOD, кој овозможува создавање на хиерархиски односи помеѓу класите. Овозможува создавање на специјализирани класи (изведени или детски класи) кои наследуваат својства и однесувања од поопшта класа (базна или матична класа). Наследувањето промовира повторна употреба на кодот, бидејќи заедничките атрибути и методи може да се дефинираат во основната класа и да се наследат од изведените класи. [31, 60, 80]

Полиморфизмот е моќен концепт во OOD кој овозможува објекти од различни класи да се третираат како примери на заедничка суперкласа. Полиморфизмот овозможува користење на апстрактни класи и интерфејси, кои дефинираат збир на методи кои изведените класи мора да ги имплементираат. Овој концепт ја олеснува флексибилноста на кодот, бидејќи дозволува различни објекти да се користат наизменично, промовирајќи растежливост и модуларност. [15, 38, 60]

Објектно-ориентираните принципи на дизајн како што се инкапсулација, наследување и полиморфизам обезбедуваат цврста основа за дизајнирање комплексни системи за игра. Тие промовираат организација на кодот, одржливост и повторна употреба, дозволувајќи им на програмерите да создадат скалабилни и робусни архитектури на игри. Со користење на овие принципи, развивачите на игри можат да се фокусираат на создавање поединечни објекти со добро дефинирани одговорности, што ќе доведе до поефикасни и модуларни бази на кодови кои се полесни за разбирање, менување и проширување во текот на процесот на развој на играта. [38, 64, 80]

Дизајн врз основа на компоненти

Дизајнот базиран на компоненти е алтернативен пристап кон развојот на игрите кој го нагласува составот на однесувањето и функционалноста преку употреба на повеќекратно употребливи и модуларни компоненти. За разлика од традиционалниот објектно-ориентиран дизајн, кој се фокусира на хиерархии на класи и наследство, дизајнот базиран на компоненти промовира создавање на ентитети составени од повеќе независни компоненти. [7, 28, 33, 76, 129]

Во дизајнот заснован на компоненти, компонентата е самостоен модул кој содржи специфична функционалност или однесување. Секоја компонента претставува посебен аспект на однесувањето на ентитетот, како што се рендерирање, физика, влез или вештачка интелигенција. Компонентите обично се дизајнирани да можат повторно да се употребуваат и може да се прикачат на различни ентитети по потреба, промовирајќи повторна употреба и модуларност на кодот. [7, 33, 76, 129]

Ентитетите во дизајнот базиран на компоненти се во суштина контејнери кои држат и управуваат со различни компоненти. Наместо да го наследат однесувањето од основната класа, ентитетите се составуваат со прикачување на потребните компоненти. Овој состав е динамичен и флексибилен, дозволувајќи им на ентитетите да имаат различни комбинации на компоненти врз основа на нивните барања. Овој пристап им овозможува на програмерите да создаваат сложени и разновидни ентитети на играта со комбинирање и конфигурирање на различни компоненти. [7, 33, 76, 129]

Една од клучните предности на дизајнот базиран на компоненти е неговата вродена флексибилност и модуларност. Компонентите може лесно да се додадат, отстранат или заменат за да го изменат однесувањето на ентитетот без да влијаат на другите компоненти или на целокупната архитектура. Оваа модуларност овозможува брзо создавање на прототипови, лесно повторување и промовира повторна употреба на кодот низ различни ентитети, па дури и различни проекти. [7, 33, 76, 129]

Друга придобивка од дизајнот базиран на компоненти е неговата способност да поддржува развој базиран на податоци. Со одвојување на однесувањето на поединечни компоненти, дизајнерите на игри и дизајнерите на нивоа можат лесно да ја дотераат и конфигурираат функционалноста на ентитетот со менување на податоците за неговите компоненти, наместо да пишуваат нов код. Оваа поделба на податоците и однесувањето ја олеснува соработката помеѓу различните улоги во развојот на играта и овозможува брзо повторување и експериментирање. [7, 33, 76, 129]

Генерално, дизајнот базиран на компоненти нуди флексибилен и модуларен пристап кон развојот на играта, фокусирајќи се на составот и повторното користење на однесувањето преку употреба на компоненти и ентитети. Обезбедува скалабилна

архитектура која промовира повторна употреба на кодот, одржливост и овозможува брзо повторување и прилагодување, што го прави популарен избор за градење сложени и динамични системи за игри. [7, 33, 76, 129]

Дизајн управуван од настани

Дизајнот управуван од настани е парадигма во развојот на играта која се врти околу концептот на настани и нивните управувачи. Се заснова на идејата дека различни делови од системот на играта комуницираат и реагираат на настани што се случуваат за време на играта. Настаните претставуваат специфични појави или промени во играта, како што се внесот од играчот, откривање на судири или промени во состојбата на играта. [17, 18, 42, 79, 126]

Во дизајнот управуван од настани, текот на играта е воден од настани, наместо следење на линеарна низа на операции. Настаните може да се генерираат и од внатрешни и од надворешни извори, и тие се испраќаат до релевантните ракувачи со настани, кои се одговорни за обработка на настаните и активирање на соодветни дејства или однесувања како одговор. [17, 18, 42, 79, 126]

Дизајнот управуван од настани промовира лабаво спојување помеѓу различни компоненти или системи во играта, бидејќи тие комуницираат преку настани наместо директни зависности. Оваа лабава спојка овозможува поголема флексибилност и модуларност, бидејќи компонентите може да се додаваат, отстрануваат или менуваат без да влијае на целиот систем. Исто така, промовира повторна употреба на кодот, бидејќи ракувачите со настани може повторно да се користат во различни делови од играта што споделуваат слични настани. [17, 18, 42, 79, 126]

Дизајнот управуван од настани овозможува пореактивно и подинамично искуство во играта. Настаните може да се обработуваат асинхроно, овозможувајќи истовремено извршување на различни дејства или однесувања како одговор на повеќе настани. Овој пристап е особено корисен за ракување со внесот, каде настаните може да се активираат со дејствата на корисникот и да се обработуваат независно, овозможувајќи одговорна и интерактивна игра. [17, 18, 42, 79, 126]

Понатаму, дизајнот управуван од настани ја олеснува екстензивноста и модификацијата. Нови настани може лесно да се додадат за да се приспособат на нови функции или интеракции, а соодветните управувачи на настани може да се развиваат одделно, интегрирајќи се беспрекорно во постоечкиот систем. Оваа флексибилност овозможува лесно проширување и модификација на играта без големо преструктуирање на кодот. [17, 18, 42, 79, 126]

На кратко, дизајнот управуван од настани обезбедува флексибилен и модуларен пристап кон развојот на играта преку искористување на моќта на настаните и нивните управувачи. Промовира лабаво спојување, реакција и повторна употреба на кодот, овозможувајќи им на програмерите да создаваат динамични и интерактивни системи за игри. Преку прифаќање на принципите за дизајн водени од настани, развивачите на игри можат да градат игри што се приспособливи, екстензивни и одговараат на активностите на играчите и промените во околината на играта.

Други модели на дизајн специфични за развојот на игри

Во развојот на игри, низ времето се појавија различни модели на дизајн за да се решат заедничките предизвици и да се олесни развојот на робусни и скалабилни системи за игри. Еве неколку модели на дизајн специфични за развојот на играта: [38, 86]

1. Јамка (Game Loop): Шемата јамка за игра обезбедува структура за контролирање на главниот тек на играта. Обично вклучува циклус кој ја ажурира состојбата на играта, обработува влез, прикажува графика и се справува со други задачи специфични за играта. Јамката обезбедува конзистентни и синхронизирани ажурирања низ различни системи на игри.

2. Состојба (State): Оваа шема се користи за управување со различни состојби или режими во играта, како што се главното мени, играњето, екранот за пауза, итн. Секоја состојба е претставена со посебен објект што ги инкапсулира однесувањето и логиката специфични за таа состојба. Овај модел ги поедноставува транзициите на состојбите и овозможува лесно додавање или менување на состојбите на играта.

3. Синглтон (Singleton): Шемата синглтон се користи за да се осигура дека постои само еден пример од класата во текот на играта. Вообичаено се користи за глобално достапни менаџери или системи, како што се менаџерот за влезови, аудио-менаџерот или управувачот со ресурси. Моделот Singleton обезбедува централизиран пристап до споделените ресурси и помага да се одржи конзистентноста и координацијата во играта.

4. Набљудувач (Observer): Оваа шема ја олеснува комуникацијата помеѓу различни компоненти на играта преку воспоставување на врска еден-на-многу. Вклучува субјект (забележлив) кој води листа на набљудувачи (претплатници) и ги известува кога ќе дојде до промена. Овај модел е корисен за системи управувани од настани, како што се известување на судири со објекти, ажурирања на резултати или настани од играта.

5. Флајвејт (Flyweight): Шемата Флајвејт се користи за оптимизирање на користењето на меморијата преку споделување заеднички податоци меѓу повеќе објекти. Во игрите, често се користи за ефикасно управување со графички или аудио ресурси. Шаблонот ги раздвојува внатрешните (споделени) и надворешните (уникатни) податоци на објектите, дозволувајќи им на повеќе објекти да споделуваат исти суштински податоци, со што се намалува меморијата.

6. Команда (Command): Оваа шема го одвојува барателот на дејство од објектот што го извршува дејството. Го инкапсулира барањето како објект, дозволувајќи му да се параметризира и да се стави во редица. Во развојот на играта, шаблонот команда може да се користи за имплементирање на управување со влезови, овозможувајќи приспособливи врзувачки копчиња и функционалност слична на макро.

Ова се само неколку примери на модели на дизајн што вообичаено се применуваат во развојот на игри. Секоја шема се справува со специфични предизвици и обезбедува структуриран пристап за решавање на заедничките проблеми, што на крајот помага во создавање на скалабилни, одржливи и ефикасни системи за игра.

II. ИСТРАЖУВАЊЕ НА UNITY

2 ЗАПОЗНАВАЊЕ СО UNITY

Овој дел го прикажува Unity погонот и неговиот интерфејс заедно со дел од многуте карактеристики.

2.1 Unity

Овој поддел ја опишува методологијата што се користи за да се запознаете со платформата за развој на игри Unity. Опфаќа теми како што се учење на интерфејсот на Unity, некои од алатките, управување со средства, состав на сцена, скриптирање и кодирање. [16, 35, 50, 52]



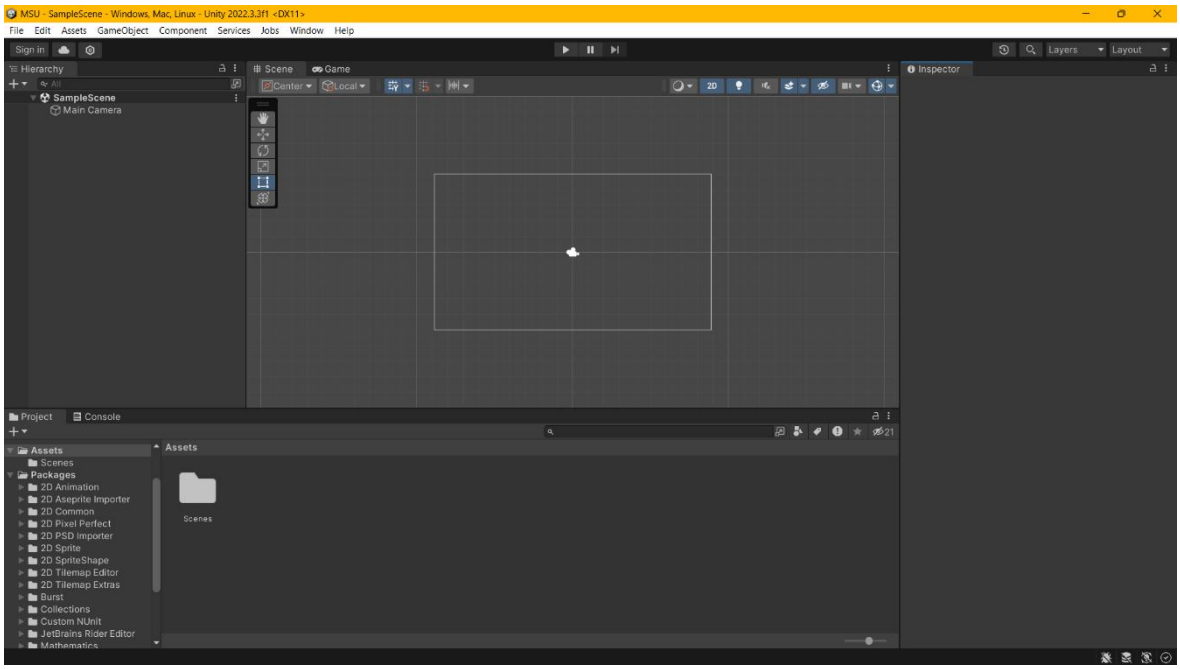
2- 1 Лого на Unity

Unity е популарен избор за развивачите на инди-игри ширум светот од 2005 година. Погоден е не само за развој на 2D и 3D игри, туку е одличен и за дизајн со виртуелна реалност (VR) и проширена реалност (AR). Погонот се ажурира секоја година со нови содржини и благодарение на големата заедница, има продавница за средства со огромна количина на бесплатни средства и средства кои се на плаќаат по употреба. Unity е бесплатен, така што секој може да почне да користи и да учи како лесно да прави игри. Популарни игри: Cuphead, Inside, Ori and the Blind Forest и многу други. Причината зошто го избрав Unity погонот во мојот труд е затоа што е погоден за правење игри самостојно. Вообичаено, од тимот се бара да направи игра затоа што има толку многу компоненти на играта што едно лице не може да се справи со сите или треба долго време да ги совладаат сите потребни вештини (анимација, звук, дизајн на околина, вештини за кодирање итн.). Unity обезбедува некои компоненти бесплатно за да може фокусот да биде на програмирањето.

2.1.1 Интерфејс на Unity

Овој пододдел се фокусира на интерфејсот Unity и различните алатки достапни во развојната околина на Unity. Обезбедува преглед на уредувачот (Editor) на Unity, вклучувајќи приказ на сцена, приказ на играта, инспектор, хиерархија и прозорец на проектот. Ги опфаќа и најчесто користените алатки како што се алатките за трансформација, поставките за внес на средства итн. Овој поддел има за цел да го запознае истражувачот со интерфејсот на Unity и алатките неопходни за развој на играта.

Корисничкиот интерфејс на Unity може да биде застрашувачки за почетниците, може да потрае некое време за да се навикне поради неговата сложеност. Во овој дел, се разгледува распоредот на интерфејсот. Може да се прилагоди распоредот на прозорците на интерфејсот на кој било начин што е погоден за нашиот проект. Го воведувам стандардниот распоред бидејќи тоа е изгледот што се гледа при креирањето на нов проект.

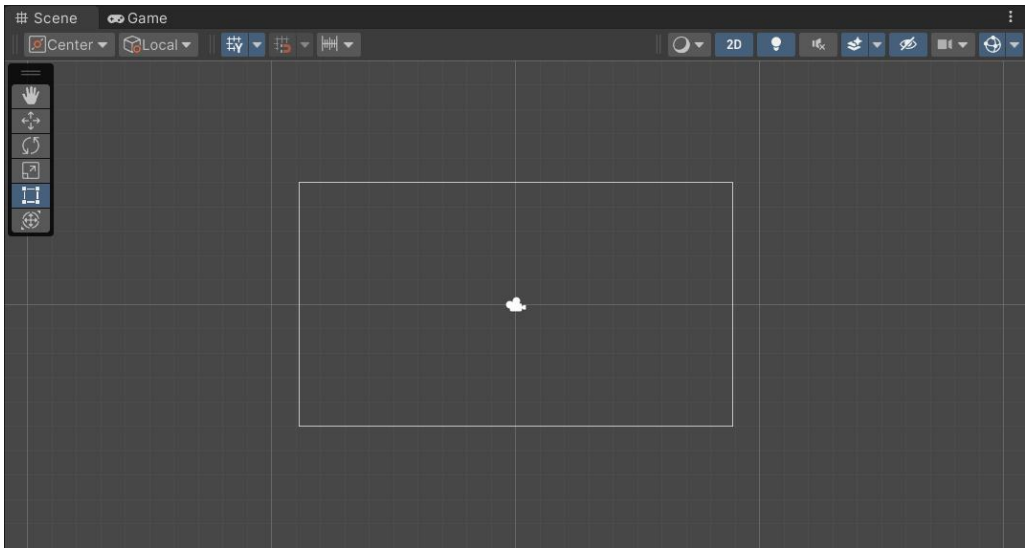


2- 2 Стандарден кориснички интерфејс на Unity

- **Поглед на сцена (Scene view)**

Во средината на интерфејсот, може да се види прозорецот на сцената. Ова е интерактивниот поглед на светот што се создава во нашата сцена. Може да се

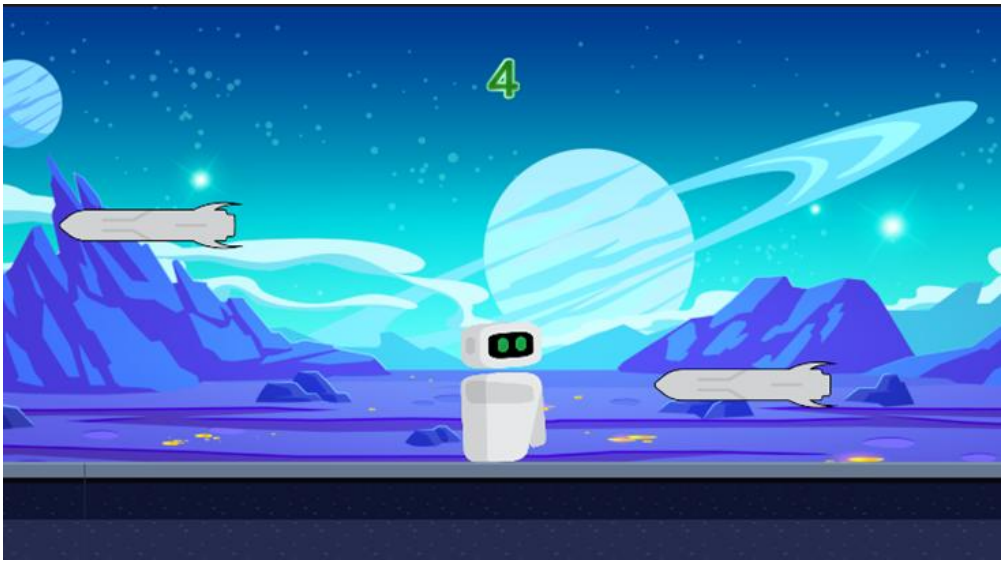
комуницира со компоненти во нашата сцена, сценографија, ликови, платформи, околини и други видови предмети за играта.



2- 3 Поглед на сцена (Scene window).

- **Поглед на играта (Game view)**

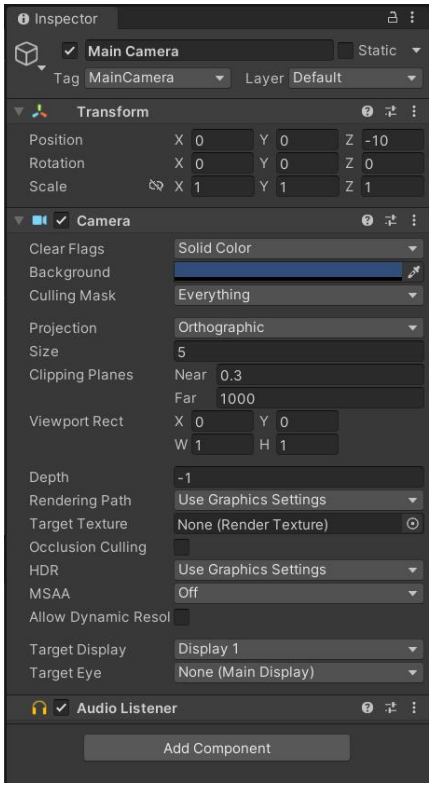
До прозорецот за сцената, е прозорецот на играта. Прозорецот на играта го визуелизира она што играчот го гледа кога ја игра играта. Најчесто не може да се комуницира со ништо во рамките на објектот на играта бидејќи неговата единствена функција е визуелизација на играта.



2- 4 Поглед на играта (Game view).

- **Инспектор**

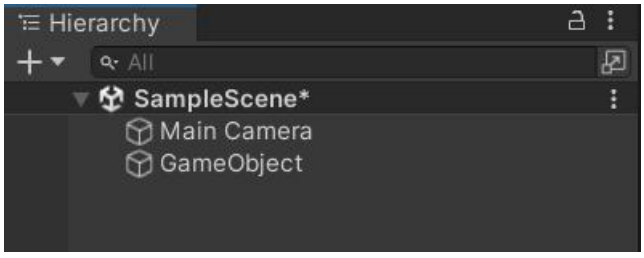
На десната страна на корисничкиот интерфејс, може да се види прозорецот Инспектор. Овој прозорец ги прикажува компонентите во нашиот моментално избран објект за игра. Исто така може да се додадат или отстранат компоненти, да се приспособат компонентите како што е соодветно за нашиот проект.



2- 5 Прозорец за инспектор на објектот на главната камера.

- **Хиерархија**

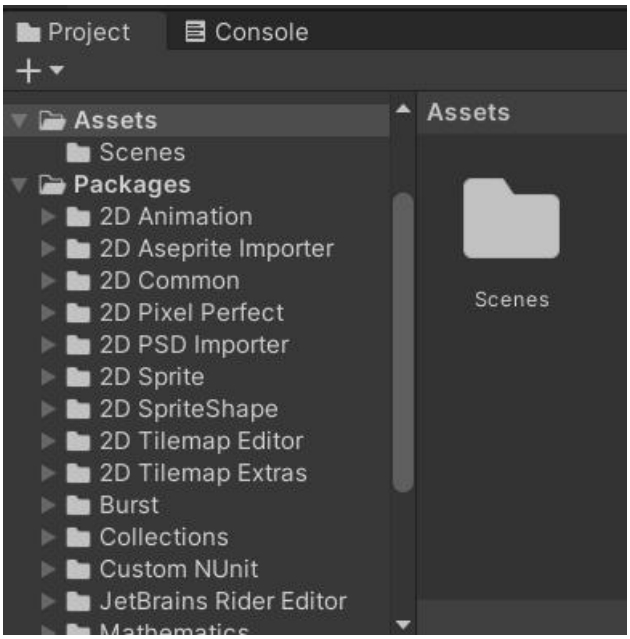
Во горниот лев агол, може да се види прозорецот Хиерархија. Го прикажува секој објект на играта што е во сцената. Што и да се додаде или отстрани во сцената, се додава или отстранува и во прозорецот за хиерархија. Објектите на играта во Unity се основни објекти кои претставуваат ликови, својства и сценографија. Тие се контејнери за компонентите да имаат смисла на играта. Празен објект на играта не прави ништо, но ако се прикачи компонента на него, тој ќе има функција. На пример, ако се додаде компонента за судир на празен објект на играта, таа сега може физички да се судри со други објекти на играта кои исто така имаат компонента за судир. Сцената ги содржи предметите на нашата игра. Сфатете ја секоја сцена како уникатно ниво на играта. Може да се создаде и главно мени каде играчот може да ја започне играта, да ги приспособи поставките и да ја напушти играта во сцената. Многу сцени се комбинираат една со друга за да направат игра.



2- 6 Прозорецот хиерархија.

- **Прозорец на проектот**

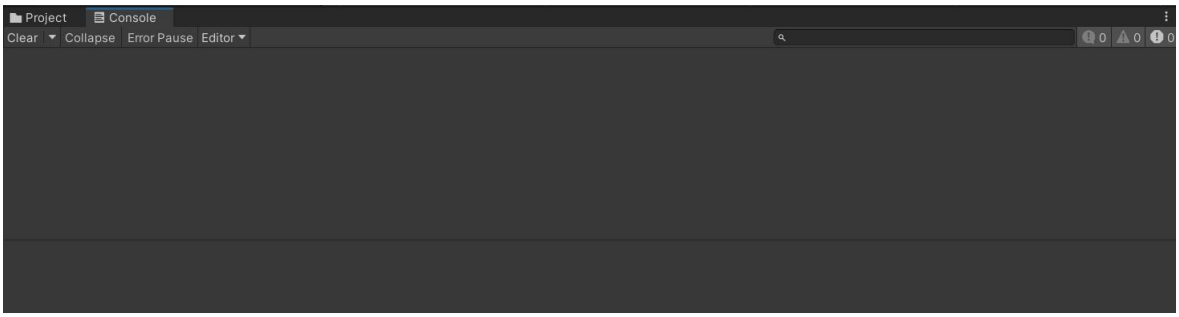
Под прозорот Хиерархија, е прозорецот Проект. Не само што ги прикажува сите датотеки од проектот на кој се работи, туку е и главниот начин за навигација и создавање нови датотеки за нашиот проект.



2- 7 Прозорец на проектот.

- **Конзола**

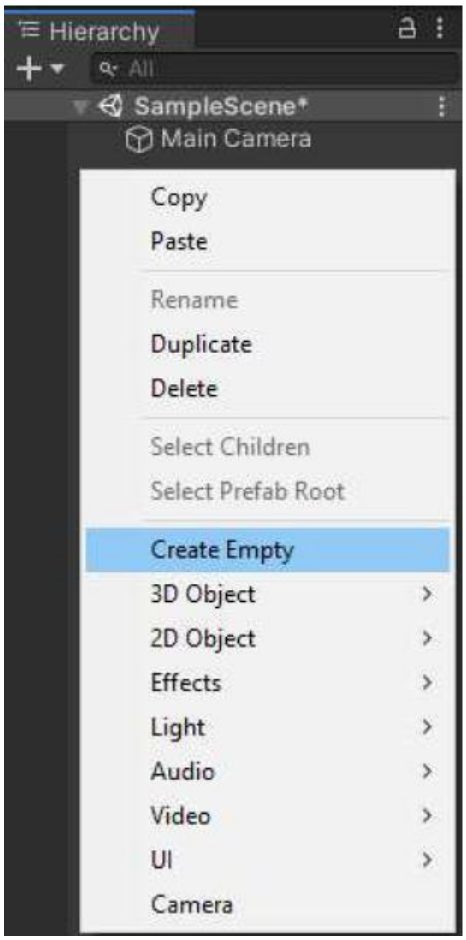
До прозорецот проект, е прозорецот конзола. Овој прозорец е еден од најважните што треба да ги има на нашиот екран, ги прикажува предупредувањата, грешките и другите пораки поврзани со нашиот проект. Овие пораки се од витално значење за нашата игра да работи без никакви проблеми. Тоа е исто така најбрзиот начин да се најдат грешки во нашиот проект.



2- 8 Прозорецот конзола.

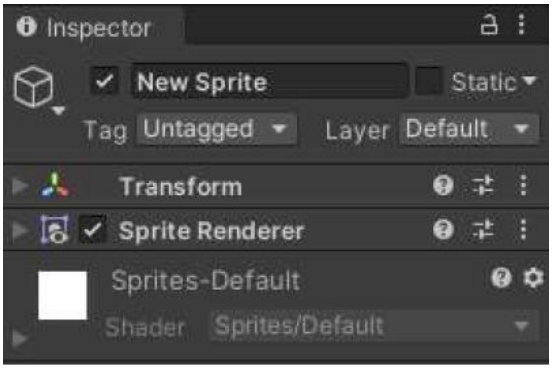
2.1.2 Основни карактеристики во Unity

Во овој дел, прикажани се некои од основните карактеристики што може да се направат со Unity. Во следното поглавје ги воведувам само карактеристиките што се користат редовно. Исто така, постојат многу начини да се направи секој од чекорите што ги воведувам овде, така што читателите можат да го истражуваат корисничкиот интерфејс и да ги најдат своите преференци.



2- 9 Креирање на празен GameObject.

За да креирате празен GameObject, едноставно кликнете со десното копче на Хиерархија и изберете „Create Empty“. Како што беше спомнато претходно, празен GameObject сè уште нема никаква функција, мора да се додадат компоненти на него. Од сликата (2-9) може да се види дека има и опции за креирање на 3D Object, 2D Object, UI објекти итн.,

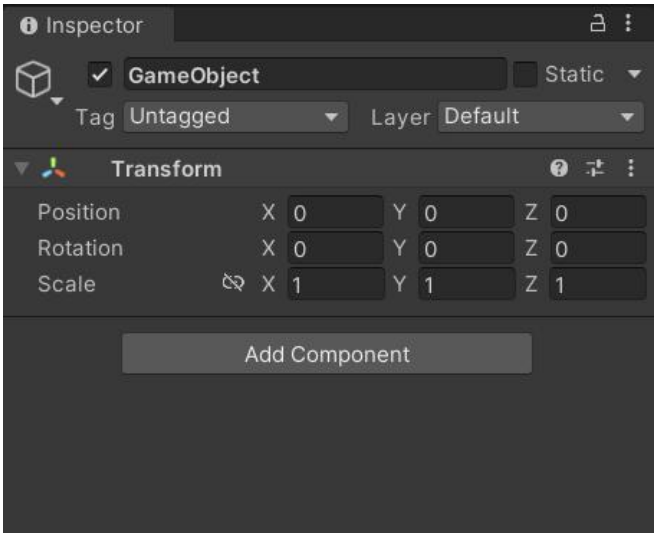


2- 10 New Sprite GameObject.

овие се исто така објекти на играта, но веќе има компоненти додадени на себе за да функционираат. На пример, ако кликнете на „2D“ и изберете „Sprite“, се создава GameObject со име „New Sprite“ со компонента „Sprite Renderer“ веќе додадена на него.

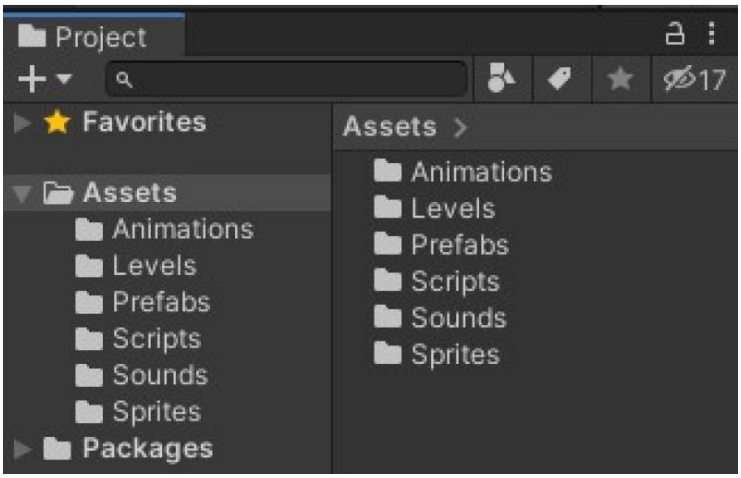
За да додадете компонента на GameObject, едноставно изберете го Game Object во прозорецот Хиерархија, а потоа во прозорецот Инспектор, можете да

кликнете на „Додај компонента (Add Component)“ и да се одбере компонентата што ни треба.



2- 11 Додавање компонента во GameObject.

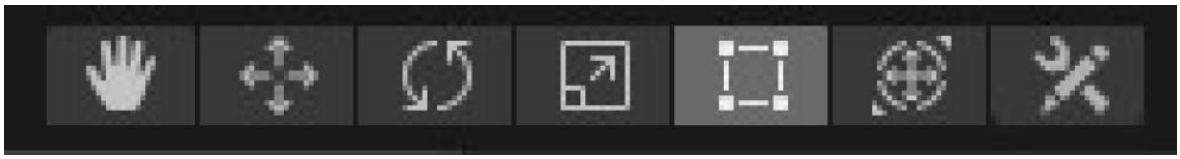
Ќе се создадат многу датотеки во нашиот проект, па затоа е од суштинско значење да се организираат датотеките во папки. Луѓето имаат различни преференции за тоа како ги организираат своите проекти, но начинот на кој го најдов ефикасен е да се организираат по типови на датотеки.



2- 12 Организирање на проект.

Како пример, начинот на кој јас се организирам на сликата погоре е според типовите на датотеки. На пример, во папката „Звук“, ги ставам сите звучни датотеки како музика во позадина, звучни ефекти. Можам да имам и подпапки за различен

тип на звуци ако имам многу звучни датотеки (гласови на знаци, музика за шефови, звуци од амбиенталниот свет, итн.). На овој начин, можам да следам што сум создал и каде можам да ги најдам.



2- 13 Лента со алатки

Лентата со алатки погоре е она што се користи за конфигурација на објектот на играта во прозорецот на сцената:

- Hand Tool: користете ја оваа алатка за да се движите низ сцената.
- Move Tool: за преместување на избраниот GameObject.
- Rotate Tool: за ротирање на избраниот GameObject.
- Scale Tool: за промена на големината на GameObject на сите оски одеднаш.
- Rect Tool: скалирање, позиција, големина и сидро за правоаголнест GameObject.

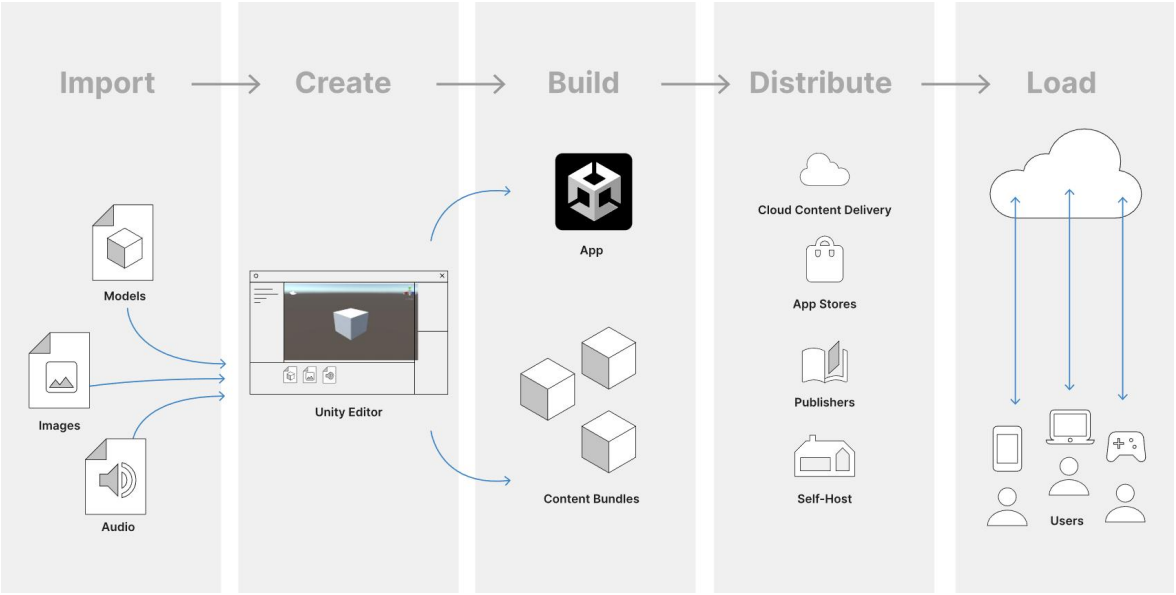
Тоа се основните карактеристики на Unity. Се разбира, има многу повеќе функции и уште повеќе начини да се прават работите со Unity, но тие постојат поради нашите преференции за тоа како сакаме да ја направиме нашата игра. Во следното поглавје, ќе воведам повеќе карактеристики на Unity бидејќи би било полесно да ги објаснам заедно со правењето на нашата прва 2D игра.

2.1.3 Управување со средства (Assets) во Unity

Овој потоддел ја истражува методологијата за управување со средства во рамките на Unity погонот. Опфаќа теми како што се внес и организирање средства, користење пакети со средства, создавање сопствени средства и оптимизирање на користењето на средствата. [88]

Средство е која било ставка што ја користите во вашиот проект за да ја креирате вашата игра или апликација. Средствата може да претставуваат визуелни или аудио елементи во вашиот проект, како што се 3D модели, текстури, sprites, звучни ефекти или музика. Средствата може да претставуваат и поапстрактни ставки како што се градиенти на бои, маски за анимација или произволен текст или нумерички податоци за каква било употреба.

Средство може да дојде од датотека создадена надвор од Unity, како што е 3D модел, аудио датотека или слика. Можете исто така да креирате некои типови средства во уредувачот Unity, како што се ProBuilder Mesh, контролер за аниматор (Animator Controller), аудио миксер или текстура на рендерирање.



2- 14 Работен тек на средствата во Unity

Горенаведениот дијаграм го прикажува типичниот работен тек кога работите со средства во Unity. Секоја колона претставува посебен чекор и е опишана подолу:

- Внесување на средства во Unity уредувачот
- Создавање на содржина со помош на Уредувачот Unity со тие средства.
- Изградување на датотека на вашата апликација или игра, а по избор и нејзините придружни пакети со содржина
- Дистрибуирање на вградените датотеки за да можат вашите корисници да пристапат до нив, преку издавач или продавница за апликации
- Вчитување на понатамошни ажурирања како што е потребно при извршување, во зависност од однесувањето на вашиот корисник и како сте ја групирале и здружиле вашата содржина.

Внес и организирање

Внесот е процес на внесување на вашите изворни датотеки во Уредувачот на Unity за работа со нив. Кога зачувувате или копирате датотека во папката Средства на вашиот проект, Unity ја внесува таа датотека што ви овозможува да работите со неа во Уредникот.

Важно е да научите некои основи за внес на средства во Unity, како на пример каде се складираат датотеките во вашиот проект, како да ги прилагодите поставките за внес за секој вид на средство, за што служат мета-датотеките и како Базата на податоци за средства ги складира внесените податоци.

Може да внесете средства создадени надвор од Unity во вашиот проект. За да го направите ова, можете или да ја изнесете датотеката директно во папката Средства за вашиот проект или да ја копирате во таа папка. За многу вообичаени формати, можете да ја зачувате вашата изворна датотека директно во папката Средства на вашиот проект и Unity може да ја чита. Unity исто така открива кога зачувувате нови промени во датотеката и повторно ги внесува датотеките по потреба.

Кога креирате Unity проект, Unity создава папка (именувана по вашиот проект) која ги содржи следните потпапки:

- Temp
- Library

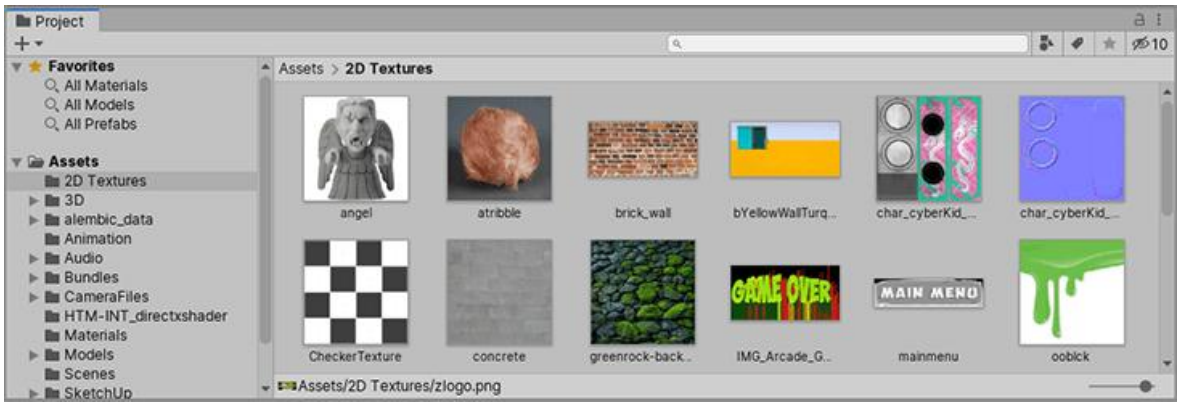
- Assets
- ProjectSettings
- Logs
- Packages

Можете да ги зачувате или копирате датотеките што сакате да ги користите во вашиот проект во папката Средства и можете да го користите прозорецот Проект во Unity за да ја прегледате содржината на вашата папка. Затоа, ако зачувате или копирате датотека во вашата папка Средства, Unity ја внесува и потоа се појавува во вашиот прозорец Проект.

Кога менувате датотека во Unity, Unity не ја менува вашата оригинална изворна датотека, иако често можете да избирате помеѓу различни начини за компресирање, менување или на друг начин да го обработите средството во Unity. Наместо тоа, процесот на внес ја чита вашата изворна датотека и создава внатрешно претставување на вашето средство подготвено за игра, што одговара на вашите избрани поставки за внес. Ако ги измените поставките за внес за средство или направите промена на изворната датотека во папката Средства, Unity повторно го внесува средството за да ги одрази вашите промени.

Предупредување: Во повеќето случаи, ставките што се појавуваат во прозорецот на вашиот проект претставуваат вистински датотеки на вашиот компјутер. Ако ги избришете во прозорецот Unity Editor Project, ќе ги избришете и од вашиот компјутер.

Unity автоматски ги открива датотеките додека се додаваат во папката Средства или ако се изменети. Кога ставате кое било средство во вашата папка Средства, тоа се појавува во прозорецот на вашиот проект.



2- 15 Прозорецот Проект ги прикажува средствата што Unity ги внесува во вашиот проект

Ако повлечете датотека од прелистувачот на датотеки на вашиот компјутер во прозорецот на Unity, Unity прави копија и ја става во вашата папка со средства. Потоа можете да пристапите до оваа копија од прозорецот Проект.

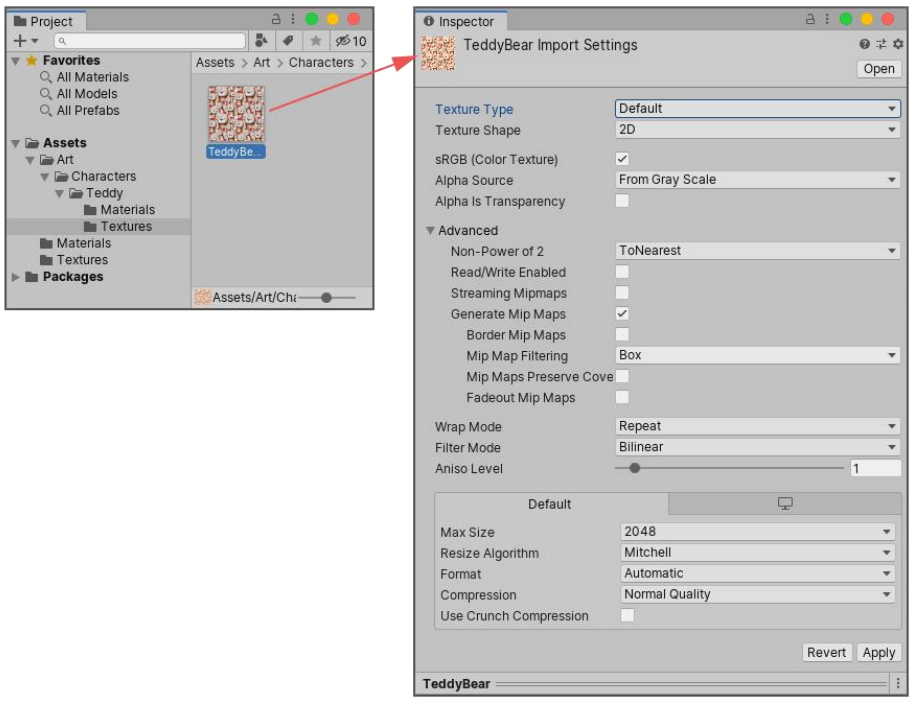
За да внесете збирки средства во вашиот проект од друг проект на Unity, можете да користите пакетите со средства.

Поставки за внес на средства

Наједноставниот начин за безбедно преместување или преименување на вашите средства е секогаш да го правите тоа од папката со проекти. На овој начин, Unity автоматски ја преместува или преименува соодветната мета-датотека.

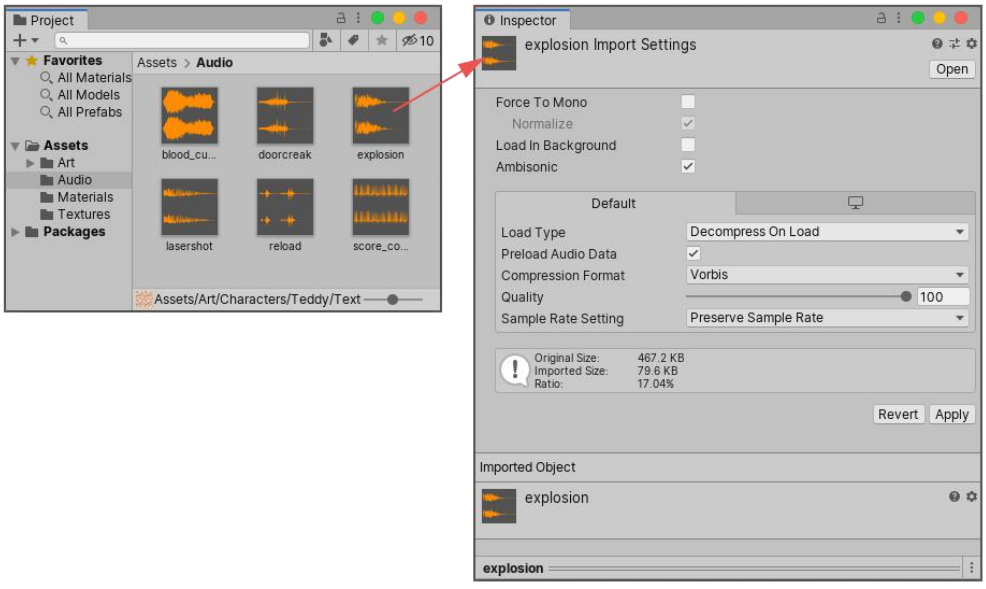
Секој тип на средство што го поддржува Unity има сет на поставки за внес, кои влијаат на тоа како се појавува или се однесува средството. За да ги видите поставките за внес на средството, изберете го средството во Проектниот приказ. Поставките за внес за ова средство ќе се појават во Инспекторот. Опциите што се појавуваат варираат во зависност од типот на избраното средство.

На пример, поставките за внес на слика ви дозволуваат да изберете дали Unity ќе ја внесе како текстура, 2D sprite или нормална мапа. Поставките за внес за датотека FBX ви дозволуваат да ја прилагодите скалата, да генерирате нормални или светлосни координати и да ги делите и скратите анимациски клипови дефинирани во датотеката.



2- 16 Со кликнување на средството со слика во прозорецот Проект се прикажуваат поставките за внес за тоа средство во Инспекторот

За други типови средства, поставките за внес изгледаат поинаку. Различните поставки што ги гледате се однесуваат на типот на избраното средство. Еве пример за аудио средство, со неговите поврзани поставки за внес прикажани во инспекторот:



2- 17 Аудио средство избрано во прозорецот Проект ги прикажува поставките за внес на аудио за тоа средство во Инспекторот

Ако развивате проект за повеќе платформи, можете да ги отфрлите „стандардните“ поставки и да доделите различни поставки за внес на основа на секоја платформа.

Обработка на средства (Asset processing)

Unity ги чита и обработува сите датотеки што ги додавате во папката Средства и ја конвертира содржината на датотеката во внатрешни податоци подготвени за игра. Самите датотеки со средства остануваат непроменети, а внатрешните податоци се зачувуваат во папката Библиотека на проектот. Овие податоци се дел од базата на податоци за средства на Unity уредникот.

Користењето внатрешни формати за средства му овозможува на Unity да има внатрешни верзии на вашите средства подготвени за употреба при извршување во Уредникот и да ги чува вашите немодифицирани изворни датотеки во папката Средства. Разликата помеѓу датотеките на средствата и внатрешните верзии значи дека можете брзо да ја уредувате датотеката со средствата и да го натерате Уредникот автоматски да ги преземе промените. На пример, можете да зачувате .psd-датотеки директно во вашата папка Средства, но хардверот како што се мобилните уреди и графичките картички на компјутерот не може директно да го обработи тој формат за да ги прикаже како текстури. Наместо тоа, Unity може да конвертира внатрешна верзија во формат што тие платформи можат да го обработат.

Unity ја складира внатрешната репрезентација на вашите средства во папката Библиотека, која се однесува како папка за кеш меморија. Како корисник, никогаш не треба да треба рачно да ја менувате папката Библиотека. Ако го правите тоа, може негативно да влијаете на вашиот проект во Уредувачот на Unity. Ова исто така значи дека не треба да ја вклучувате папката Библиотека под контрола на верзијата.

Забелешка: Ако вашиот проект не е отворен во Unity, можете безбедно да ја избришете папката Библиотека, бидејќи Unity може да ги регенерира сите негови податоци од папките Средства и Поставки на Проектот, следниот пат кога ќе го стартувате вашиот проект.

Комплексни средства

Во некои случаи, Unity може да создаде повеќе средства додека внесува една датотека со средства. На пример:

Кога 3D-датотеката (како што е датотеката FBX) дефинира материјали или содржи вградени текстури. За да се справи со ова, Unity ги извлекува материјалите и вградените текстури како посебни средства.

Кога сакате да внесете датотека со слика како повеќе 2D sprites. Треба да го користите 2D Sprite Editor за да дефинирате повеќе sprite од една графичка слика. Unity го прикажува секој sprite дефиниран во Уредникот како посебен Sprite средство во прозорецот Проект.

Кога 3Д-датотеката содржи повеќе временски линии за анимација или повеќе клипови. За да се справи со ова, Unity автоматски дефинира посебни временски линии или анимациски клипови врз основа на неговите поставки за внес на анимација. Добиените повеќекратни анимациски клипови се појавуваат како посебни средства за анимациски клипови во прозорецот на проектот.

Создавање сопствени средства

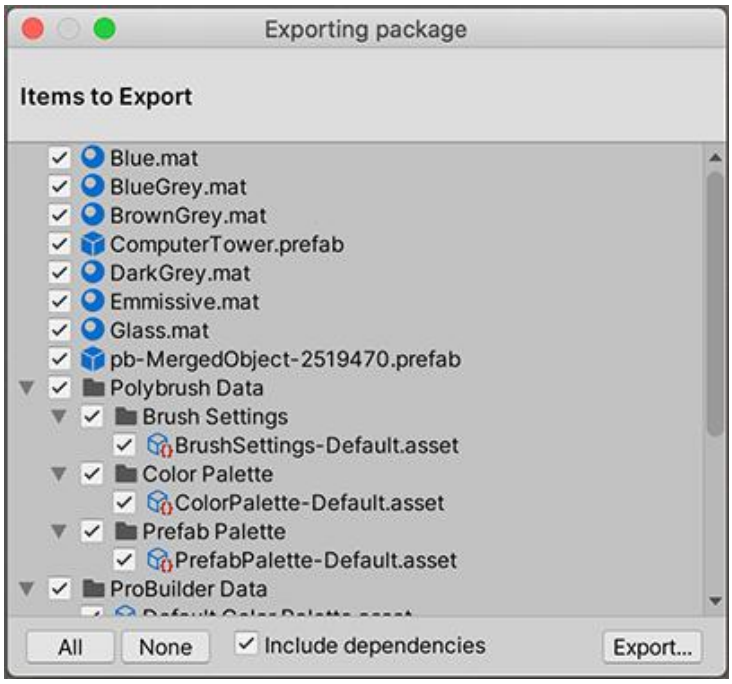
Забелешка: Ако сакате да споделувате сцени, примероци, алатки или други средства, Unity препорачува да креирате сопствен Unity пакет. Сепак, можете да го користите и форматот на пакетот средства (.unitypackage). На пример, можеби ќе сакате да креирате пакет на средства за да копирате голем број средства или цела сцена од еден проект во друг.

Следете ги инструкциите во овој дел за изнес на пакети со средства од менито Средства во Уредувачот на Unity.

Забелешка: Ако вашите средства се со висок квалитет и мислите дека на другите корисници би им биле корисни, следете ги упатствата во документацијата за објавување во продавницата за средства за да креирате нацрт пакет и да го поставите во продавницата за средства.

За да креирате сопствен пакет со сопствени средства:

1. Отворете го проектот од кој сакате да ги изнесете средствата.
2. Изберете Средства > Изнеси пакет, од менито за да се отвори прозорецот за прозорецот за Изнесување на пакети.



2- 18 Прозорец за изнес на пакет

3. Во прозорот, изберете ги средствата што сакате да ги вклучите во пакетот со кликнување на полињата за да бидат проверени.
4. Оставете ги вклучени зависностите овозможени за автоматско избирање на сите средства што се користат од средствата што веќе сте ги избрале.
5. Кликнете Изнес за да се појави истражувачот на датотеки и изберете каде сакате да ја зачувате датотеката со пакетот.
6. Именувајте и зачувајте го пакетот каде што сакате.

Совет: При изнесување на пакет, Unity може да ги изнесува и сите зависности. Така, на пример, ако изберете Сцена и изнесете пакет со сите зависности, тогаш Unity ги изнесува сите модели, текстури и други средства што се појавуваат и во сцената. Ова може да биде брз начин за изнес на неколку средства без рачно лоцирање на сите.

Забелешка: Ако изберете вклучени зависности и вашиот моментален избор содржи или упатува на какви било скрипти, Unity ги вклучува сите скрипти во

вашиот проект. Ова е затоа што Unity нема начин да утврди дали скриптите во вашиот избор упатуваат на други скрипти. На пример, ако изберете средство Сцена во вашиот проект за изнесување, а таа сцена содржи GameObject со приложена скрипта, тогаш Unity ги брои сите скрипти во вашиот проект како зависност. Ако не сакате Unity да го прави ова, треба да го отштиклирате вклучени зависности.

Стратегии за именување

Преименувајте го ажурираниот пакет користејќи поединечни имиња: на пример, МоиПакетиСредства1, МоиПакетиСредства2. Unity го препознава како ажурирање, па затоа треба да користите конвенција за именување што е јасна за вас и за сите со кои ја споделувате.

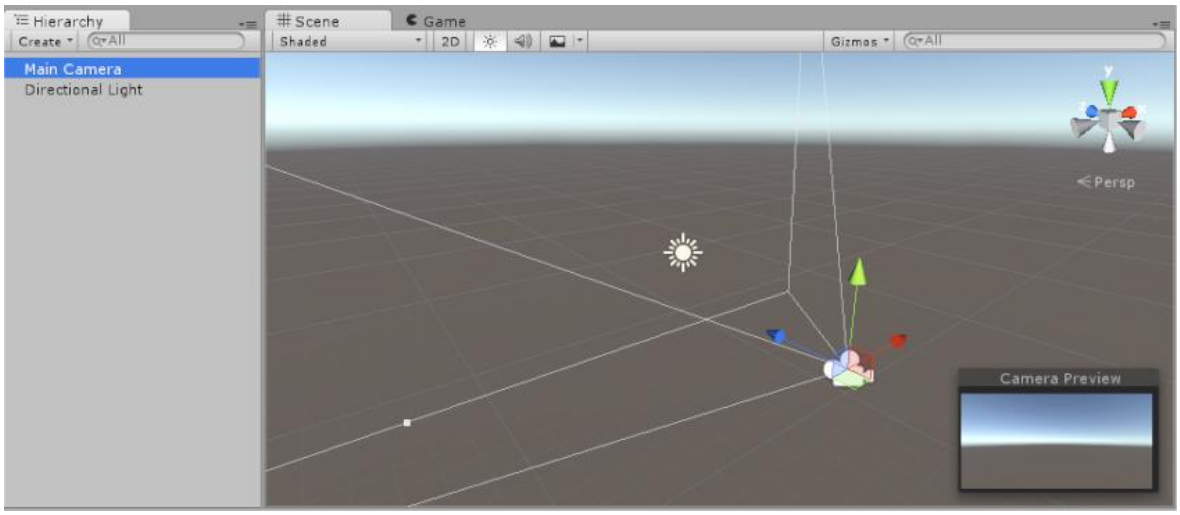
Предупредување: не отстранувајте датотеки од пакетите со средства и потоа додавајте различни датотеки со исто име. Unity користи уникатни идентификатори за следење на датотеките, така што ги препознава како различни и можеби конфликтни датотеки. Во овие случаи, Unity прикажува предупредувачки симбол кога ги внесува. Ако сте отстраниле датотека, а потоа одлучиле да ја замените, преименувајте ја на друго, дури и ако е блиску до оригиналното име.

2.1.4 Композиција на сцена во Unity

Оваа потсекција го истражува процесот на композиција на сцена и анимација во Unity. разговара за техники за креирање и уредување сцени, поставување камери, додавање осветлување и сенки.

Сцените се каде што работите со содржината во Unity. Тие се средства што содржат цела или дел од играта или апликацијата. На пример, може да изградите едноставна игра во една сцена, додека за покомплексна игра, може да користите една сцена по ниво, секоја со свои средини, ликови, пречки, украси и интерфејс. Можете да креирате кој било број на сцени во проект.

Кога креирате нов проект и го отворате за првпат , Unity отвора примерок од сцената што содржи само камера и светлина.



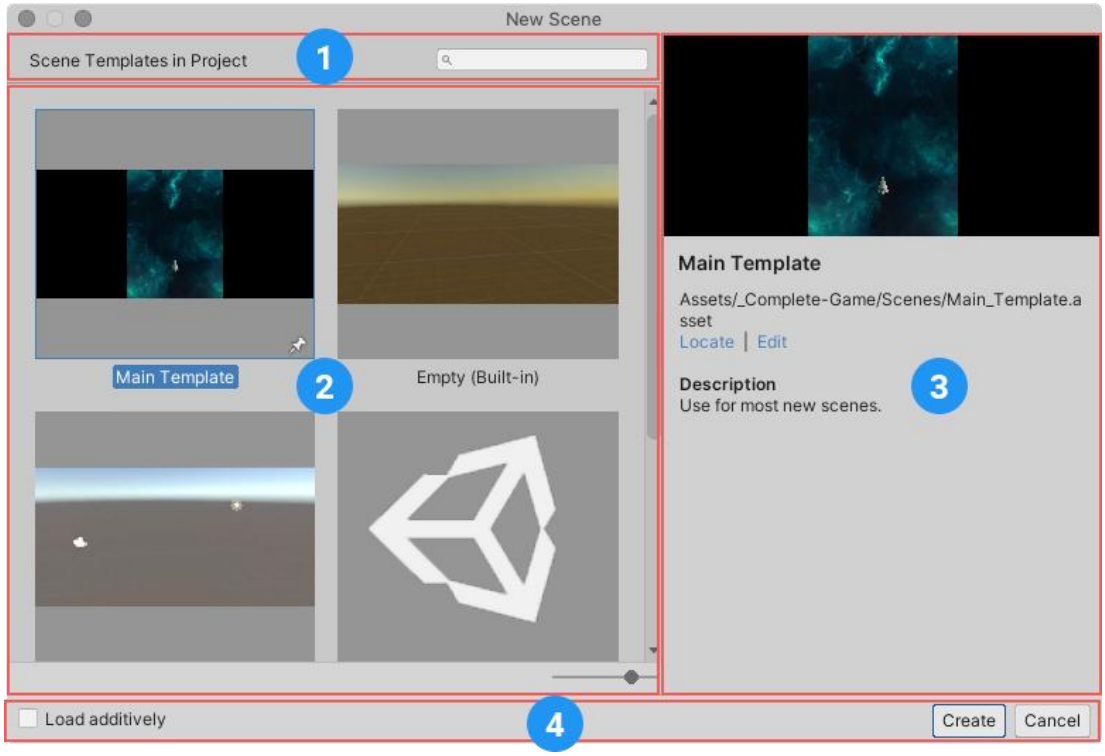
2- 19 Нова празна сцена, со стандардни 3D објекти - камера и насочено светло.

Шаблони за сцени (Templates)

Unity користи шаблони за сцени за да креира нови сцени. Шаблони за сцени се средства што се складирани во проект. Тие се слични на сцените, но се дизајнирани да се копираат наместо директно да се користат.

Прозорецот Нова сцена

Прозорецот Нова сцена се отвора кога креирате нова сцена од менито Датотека: (Датотека > Нова сцена) или кратенката Ctrl/Cmd + n. Користете го за да креирате нови сцени од одредени шаблони за сцени во вашиот проект и да добиете информации за постоечките шаблони.

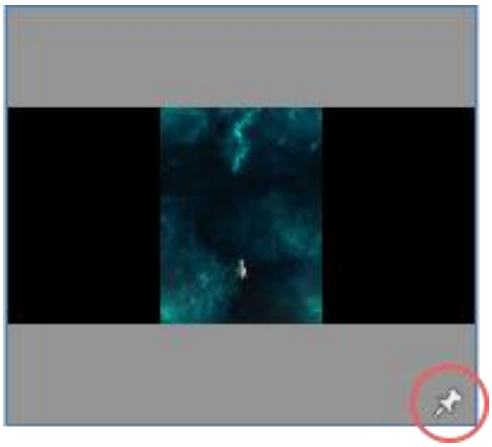


2- 20 The New Scene dialog

1. Поле за пребарување: наоѓање на достапни шаблони за сцени по име.
2. Шаблони: список на сите достапни шаблони во проектот.
3. Детали за шаблон: прикажува информации за моментално избраниот шаблон.
4. Командна лента: обезбедува команди и опции за креирање нова сцена од избраниот шаблон.

Закачување на шаблони

Закачените шаблони се појавуваат пред другите шаблони во списокот со шаблони на прозорецот Нова сцена. Последниот закачен шаблон се појавува на врвот на листата. Кликнете на иконата игла на кој било шаблон за да ја прикачите.

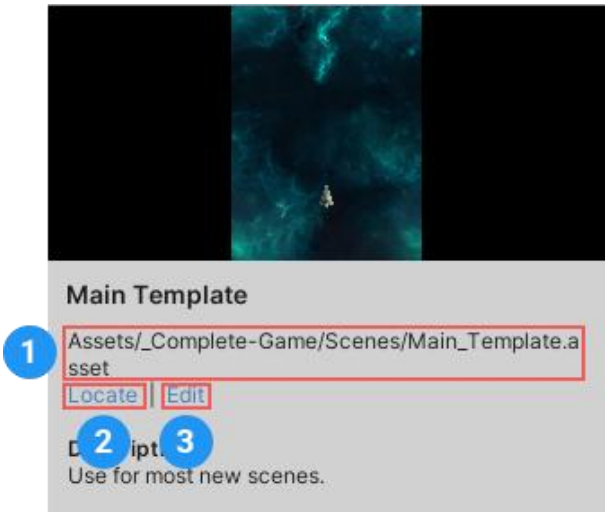


2- 21 Закачување на шаблон

Можете исто така да закачите шаблон кога ги уредувате неговите својства. Во шаблонот за сцена „Инспектор“, овозможете ја опцијата „Закачи“ во прозорецот со нова сцена.

Лоцирање и уредување шаблони

Кога ќе изберете шаблон во прозорецот Нова сцена, панелот со детали ја прикажува патеката до шаблонот (1 на сликата подолу). За да го означите шаблонот во прозорецот Проект, изберете ја врската Лоцирај (2 на сликата подолу). За да го отворите шаблонот во прозорецот на Инспекторот и да ги уредите неговите својства, изберете ја врската Уреди (3 на сликата подолу).



2- 22 Лоцирање и уредување на шаблон

Креирање и уредување сцени

Постојат неколку начини да се создаде нова сцена:

- Користејќи го прозорецот Нова сцена за да креирате нова сцена од одреден шаблон за сцена.
- Користејќи го менито или прозорецот Проект за да креирате нови сцени од шаблонот за основна сцена на вашиот проект без да го отворите прозорецот Нова сцена.
- Креирање сцена од одреден шаблон директно од скрипта на C#.

Креирање нова сцена од прозорецот Нова сцена

Користете го прозорецот Нова сцена за да креирате нови сцени од одредени шаблони за сцени во вашиот проект. Можете исто така да го користите прозорецот Нова сцена за да најдете и управувате со шаблони за сцени. Стандардно, прозорецот Нова сцена се отвора кога креирате нова сцена од менито (Датотека > Нова сцена) или со помош на кратенка (Ctrl/Cmd + n).

За да креирате нова сцена:

1. Изберете образец од списокот.
2. Ако сакате Unity дополнително да ја вчита новата сцена, овозможете Load Additively.
3. Кликнете Креирај за да креирате нова сцена од шаблонот.

Ако шаблонот нема никакви зависности што може да се клонираат, Unity ја вчитува новата сцена во меморијата, но не ја зачувува.

Ако шаблонот има зависности што може да се клонираат, Unity ве поттикнува да изберете локација во Проектот за да ја зачувате. Кога ја зачувувате сцената, Unity создава папка на истата локација и со истото име како и новата сцена. Потоа ги

клонира зависностите што може да се клонираат во новата папка и ја ажурира новата сцена за да ги користи клонираните средства наместо оригиналните средства што ги користи сцената на шаблонот.

Креирање нова сцена од менито:

Користете го менито (Средства > Креирај > Сцена) за да креирате нова сцена без да го отворите прозорецот Нова сцена.

Кога креирате нова сцена од менито, Unity автоматски го копира Основниот шаблон на проектот и ја додава новата сцена во која било папка моментално отворена во прозорецот на проектот.

Креирање нова сцена од прозорецот на проектот

Користете го контекстното мени во прозорецот Проект за да креирате нова сцена без да го отворите прозорецот Нова сцена.

1. Одете до папката каде што сакате да ја креирате новата сцена.
2. Кликнете со десното копче на папката во левиот панел или кликнете со десното копче на празно место во десниот панел и изберете Креирај > Сцена од контекстното мени.

Кога креирате нова сцена од менито, Unity автоматски го копира Основниот шаблон на проектот и ја додава новата сцена во избраната папка.

Создавање нова сцена од скрипта на C#

За да креирате нова сцена од скрипта на C# користејќи специфичен шаблон за сцена, користете го методот Instantiate.

Tuple<Scene, SceneAsset> SceneTemplate.Instantiate(SceneTemplateAsset sceneTemplate, bool loadAdditively, string newSceneOutputPath = null);

Методот Instantiate инстанцира нова сцена од шаблон за сцена. Ја враќа новосоздадената рачка Сцена и нејзините соодветни Средства на сцена. Можете да ја креирате оваа сцена дополнително. Ако сцената содржи средства што треба да се клонираат, мора да обезбедите патека за Unity да ја зачува сцената на дискот.

Нови сцени на настани

Кога креирате нова сцена од шаблон, или од скрипта или користејќи го прозорецот Нова сцена, Unity активира настан. Unity го активира овој настан откако ќе се инстанцира шаблонот, а исто така и откако ќе ги активира настаните EditorSceneManager.newSceneCreated или EditorSceneManager.sceneOpened.

```
public class SceneTemplate

{

   public delegate void NewTemplateInstantiated(SceneTemplateAsset
sceneTemplateAsset, Scene scene, SceneAsset sceneAsset, bool additiveLoad);

   public static event NewTemplateInstantiated newSceneTemplateInstantiated;

}
```

Вчитување на сцени

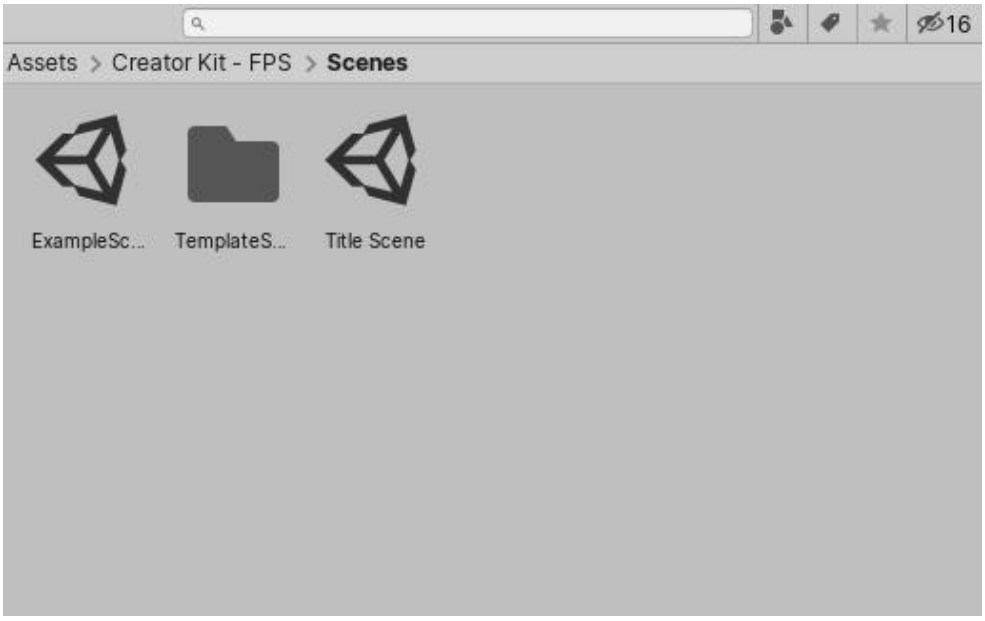
За да отворите сцена, направете едно од следново:

- Во прозорецот Проект, кликнете двапати на средствата на сцената.
- Од менито, изберете Датотека > Нова сцена
- Од менито, изберете Датотека > Неодамнешни сцени > [Име на сцената]

Ако вашата тековна сцена содржи незачувани промени, Unity ве поттикнува да ја зачувате сцената или да ги отфрлите промените.

Зачувување на сцените

За да ја зачувате сцената на која моментално работите, изберете Датотека > Зачувај сцена од менито или притиснете Ctrl + S (Windows) или Cmd + S (macOS).



2- 23 Зачуваните средства на сцената видливи во прозорецот на проектот

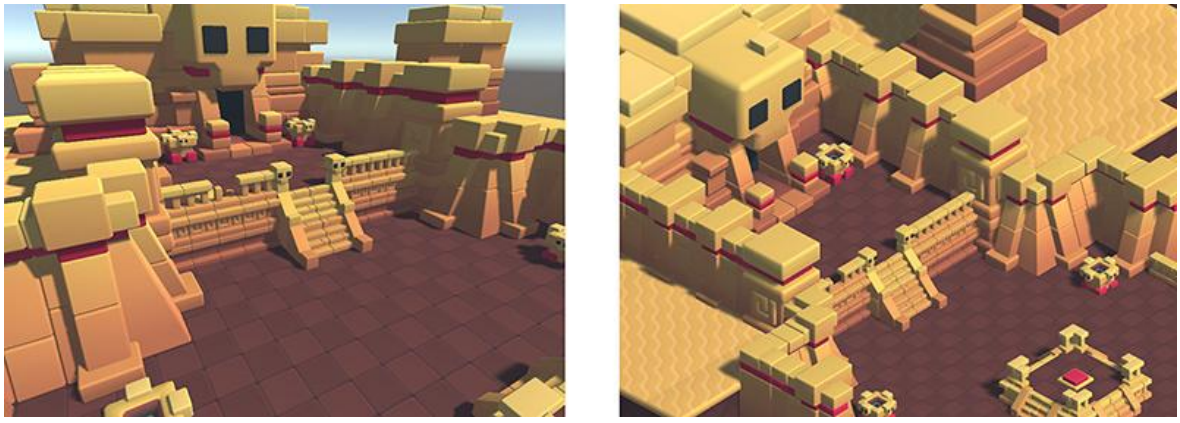
Поставување камера

Сцената во Unity ги претставува објектите во тродимензионален простор. Бидејќи екранот на гледачот е дводимензионален, Unity треба да сними поглед и да го „израмни“ за прикажување. Тоа го прави со помош на камери. Во Unity, креирате камера со додавање на компонента Камера на Game Object.

Дефинирање што гледа камерата

Она што камерата го гледа е дефинирано со неговата трансформација и нејзината компонента на камерата. Позицијата на трансформацијата ја дефинира гледната точка, (Z) оската ја дефинира насоката на погледот, а (Y) оската го дефинира горниот дел на екранот. Поставките на компонентата Камера ја дефинираат големината и обликот на областа што спаѓа во приказот. Со поставување на овие параметри, камерата може да го прикаже она што моментално го „гледа“ на екранот. Како што објектот се движи и ротира, прикажаниот приказ се движи и ротира соодветно.

Перспективна и ортогонална камера



2- 24 Истата сцена прикажана во режим на перспектива (лево) и ортогонален режим (десно)

Камерата во реалниот свет, или човечкото око, го гледа светот на начин што прави предметите да изгледаат помали колку што се подалеку од гледна точка. Овој

добро познат перспективен ефект е широко користен во уметноста и компјутерската графика и е важен за создавање реална сцена. Секако, Unity поддржува перспективни камери, но за некои цели, сакате да го прикажете приказот без овој ефект. На пример, можеби ќе сакате да креирате мапа или информативен приказ што не би требало да изгледа точно како објект од реалниот свет. Камерата која не ја намалува големината на објектите со растојание се нарекува ортогонална проекција, а Unity камерите исто така имаат опција за ова. Перспективниот и ортогоналниот начин на гледање сцена се познати како проекции на камерата.

Обликот на гледаниот регион

И перспективните и ортогоналните камери имаат ограничување колку далеку можат да „видат“ од нивната моментална позиција. Границата е дефинирана со рамнина што е нормална на насоката нанапред (Z) на камерата. Ова е познато како далечна рамнина на клипинг бидејќи предметите на поголема оддалеченост од камерата се „пресечени“ (т.е. исклучени од рендерирање). Исто така, постои соодветна рамнина за блиску до клипирање блиску до камерата - опсегот на растојание што може да се гледа е оној помеѓу двете рамнини.

Без перспектива, предметите се појавуваат со иста големина без оглед на нивната оддалеченост. Ова значи дека обемот на гледање на ортогоналната камера е дефиниран со правоаголна кутија што се протега помеѓу двете рамнини за клипирање.

Кога се користи перспективната, се чини дека предметите се намалуваат во големина како што се зголемува растојанието од камерата. Ова значи дека ширината и висината на видливиот дел од сцената растат со зголемување на растојанието. Волуменот на гледање на перспективната камера, значи, не е кутија, туку пирамидална форма со врвот на позицијата на камерата и основата на далечната рамнина на клипирање. Обликот не е точно пирамида, сепак, бидејќи врвот е отсечен од блиската рамнина за сечење; овој вид на скратена пирамида е познат како фрустум. Бидејќи неговата висина не е константна, фрустумот се дефинира со односот на неговата ширина и неговата висина (познат како сооднос на аспект) и аголот помеѓу горниот и долниот дел на врвот (познат како видно поле или FOV).

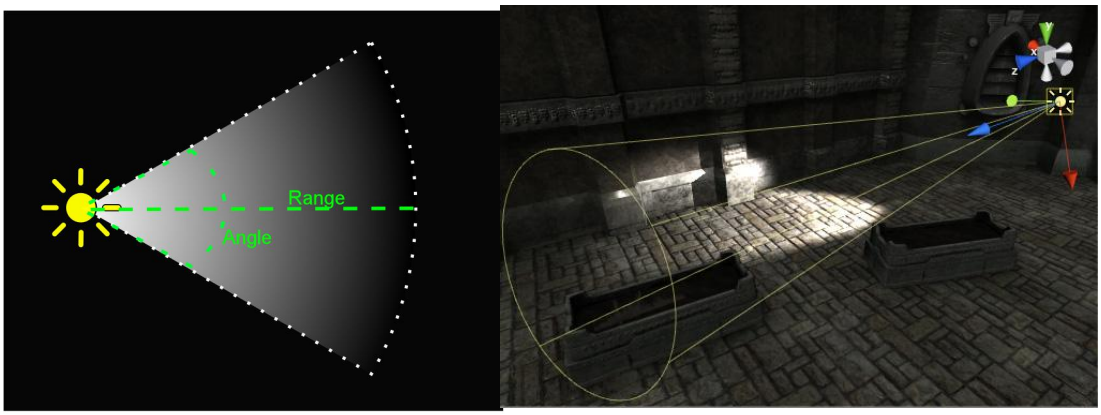
Позадина на приказот на камерата

Можете да поставите што прави камерата со позадината што ја гледате во празните области помеѓу објектите, пред да ја прикаже сцената.

На пример, можете да изберете да ја пополните заднината со рамна боја пред да ја рендерирате сцената над неа, или да нацртате небо или далечна позадина, па дури и да ја оставите таму содржината на претходната рамка.

Додавање на осветлување и сенки

Осветлувањето во Unity работи со приближување на тоа како светлината се однесува во реалниот свет. Unity користи детални модели за тоа како светлината работи за пореален резултат или поедноставени модели за постилизиран резултат.



2- 25 Ефект на осветлено место во сцената

Директно и индиректно осветлување

Директната светлина е светлина што се емитува, еднаш удира во површина и потоа се рефлектира директно во сензор (на пример, мрежницата на окото или камерата). Индиректната светлина е целата друга светлина што на крајот се рефлектира во сензорот, вклучувајќи ја и светлината што удира во површините неколку пати. За да постигнете реални резултати на осветлувањето, треба да симулирате и директна и индиректна светлина.

Unity може да пресмета директно осветлување, индиректно осветлување или, и директно и индиректно осветлување. Техниките за осветлување што ги користи Unity зависат од тоа како го конфигурирате вашиот проект.

Осветлување во реално време и печено осветлување

Осветлувањето во реално време е кога Unity го пресметува осветлувањето при извршување. Печеното осветлување е кога Unity однапред врши пресметки за осветлувањето и ги зачувува резултатите како податоци за осветлување, кои потоа се применуваат при извршување. Во Unity, вашиот проект може да користи осветлување во реално време, печено осветлување или мешавина од двете (наречено мешано осветлување).

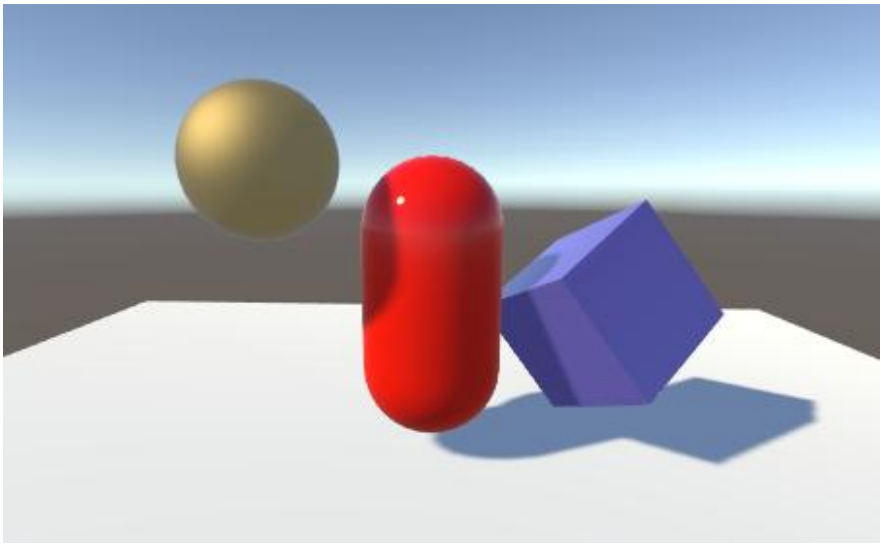
Глобално осветлување

Глобалното осветлување е група техники кои моделираат директно и индиректно осветлување за да обезбедат реални резултати на осветлување. Unity има два глобални системи за осветлување, кои комбинираат директно и индиректно осветлување.

Системот Baked Global Illumination се состои од светлосни мапи, светлосни сонди и сонди за рефлексија. Можете да печете со Progressive Lightmapper (CPU или GPU) или Enlighten Baked Global Illumination. Сепак, Enlighten Baked Global Illumination е застарено и повеќе не е видлив во корисничкиот интерфејс.

Сенки

Сенките додаваат одреден степен на длабочина и реализам на сцената бидејќи ги истакнуваат размерите и положбата на објектите кои инаку би можеле да изгледаат рамни. Во Unity, Светлината може да фрла сенки од Објектот на други делови од себе или на други блиски објекти.



2- 26 Сцена со предмети кои фрлаат сенки

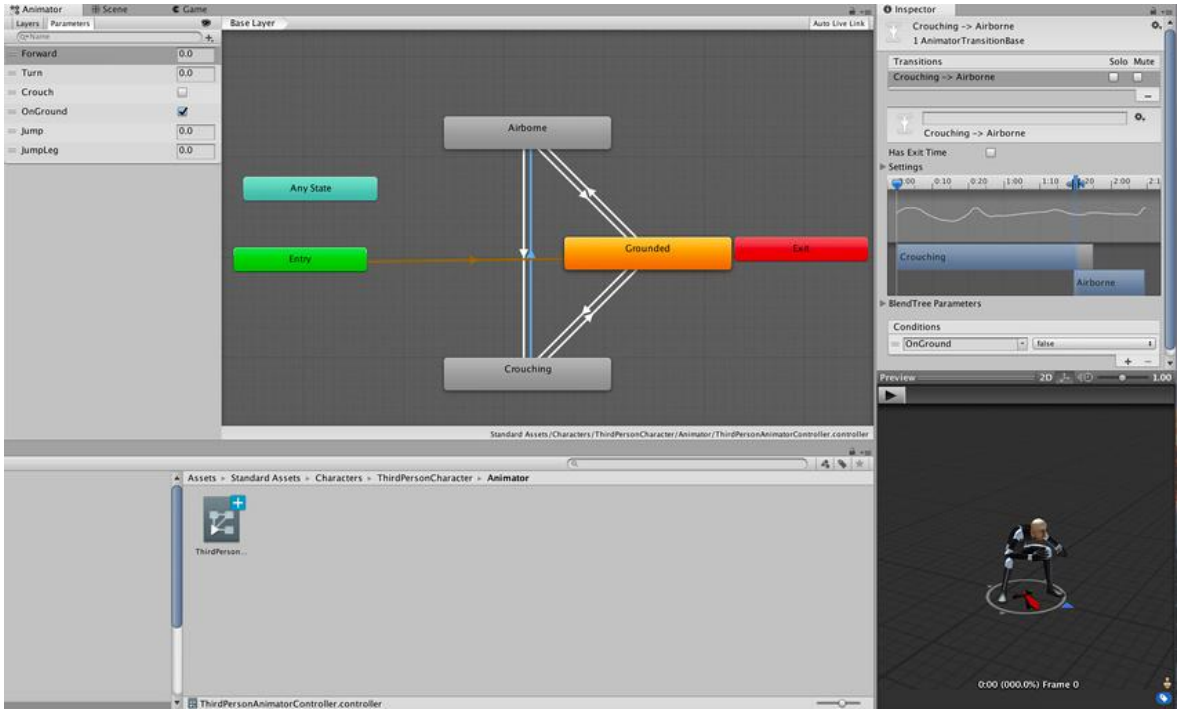
Имплементација на анимации

Unity има богат и софистициран систем за анимација (понекогаш се нарекува „Mecanim“). Обезбедува:

- Лесен работен тек и поставување на анимации за сите елементи на Unity вклучувајќи објекти, знаци и својства.
- Поддршка за внесени анимациски клипови и анимација креирана во Unity
- Ретаргетирање на човеколикна анимација - можност за примена на анимации од еден модел на карактери на друг.
- Поедноставен работен тек за усогласување на анимациските клипови.
- Практичен преглед на анимациските клипови, транзициите и интеракциите меѓу нив. Ова им овозможува на аниматорите да работат понезависно од

програмерите, да ги прототипираат и прегледуваат нивните анимации пред да се закачи кодот за играње.

- Управување со сложени интеракции помеѓу анимации со алатка за визуелно програмирање.
- Анимирање на различни делови од телото со различна логика.
- Карактеристики на слоеви и маскирање.



2- 27 Типичен приказ на машина за состојба на анимација во прозорецот на аниматорот

Работен тек за анимација

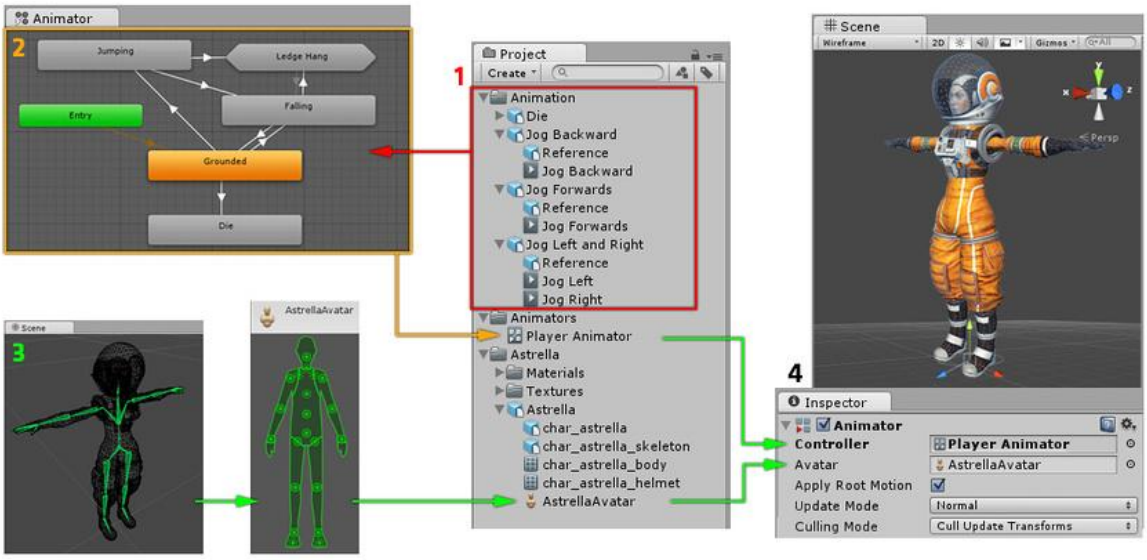
Системот за анимација на Unity се заснова на концептот на анимациски клипови, кои содржат информации за тоа како одредени објекти треба да ја променат својата позиција, ротација или други својства со текот на времето. Секој клип може да се смета како единствена линеарна снимка. Анимациските клипови од надворешни извори се креирани од уметници или аниматори со алатки од трета страна, како што се Autodesk 3ds Max или Autodesk Maya, или доаѓаат од студија за снимање движења или други извори.

Потоа, анимациските клипови се организираат во структуриран систем на дијаграм на текови наречен контролер на аниматори. Контролерот на аниматорот делува како „машина на состојба“ која следи кој клип треба да се репродуцира во моментот и кога анимациите треба да се менуваат или да се спојат.

Многу едноставен контролер за аниматор може да содржи само еден или два клипови, на пример за контрола на вртењето и потскокнувањето или за анимирање на отворање и затворање на вратата во точното време. Понапредниот контролер за аниматори може да содржи десетици човеколикни анимации за сите дејства на главниот лик и може да се спојува помеѓу повеќе клипови во исто време за да обезбеди течно движење додека играчот се движи низ сцената.

Системот за анимација на Unity, исто така, има бројни специјални карактеристики за ракување со човеколикни ликови кои ви даваат можност да ја пренасочите човеколикната анимација од кој било извор (на пример: снимање на движења; продавница за средства; или некоја друга библиотека за анимации од трета страна) до вашиот сопствен модел на карактери. Овие посебни карактеристики се овозможени од системот Аватар на Unity, каде што човеколикните знаци се мапирани во заеднички внатрешен формат.

Секое од овие парчиња - анимациски клипови, контролорот на аниматорот и аватарот, се споени на GameObject-от преку компонентата Аниматор. Оваа компонента има референца за контролер на аниматор и (ако е потребно) аватар за овој модел. Контролорот за анимација, пак, ги содржи референците за анимациските клипови што ги користи.



2- 28 Дијаграм што покажува како различни делови од системот за анимација се поврзуваат заедно

Горенаведениот дијаграм го покажува следново:

1. Анимациските клипови се внесуваат од надворешен извор или се создаваат во Unity. Во овој пример, тие се внесени човеколикни анимации снимени со движење.
2. Анимациските клипови се поставени и распоредени во контролер за аниматор. Ова покажува приказ на контролер за аниматор во прозорецот Аниматор. Состојбите (кои може да претставуваат анимации или вгнездени машини за потсостојби) се појавуваат како јазли поврзани со линии. Овој контролер за аниматор постои како средство во прозорецот на проектот.
3. Моделот на монтирани знаци (во овој случај, астронаутот „Астрела“) има специфична конфигурација на коски кои се мапирани на заедничкиот формат на Аватар на Unity. Ова мапирање е зачувано како средство на Аватар како дел од внесениот модел на знаци, а исто така се појавува во прозорецот Проект како што е прикажано.
4. При анимирање на моделот на знаци, има прикачена компонента Аниматор. Во приказот Инспектор прикажан погоре, можете да ја видите компонентата Аниматор на која се доделени и контролорот на аниматорот и аватарот. Аниматорот ги користи овие заедно за да го анимира моделот. Референцата

Аватар е неопходна само кога се анимира човеколикен лик. За други типови на анимација, потребен е само контролер за аниматор.

Криви на анимација

Можете да прикачите криви на анимација на внесените анимациски клипови во картичката Анимација.

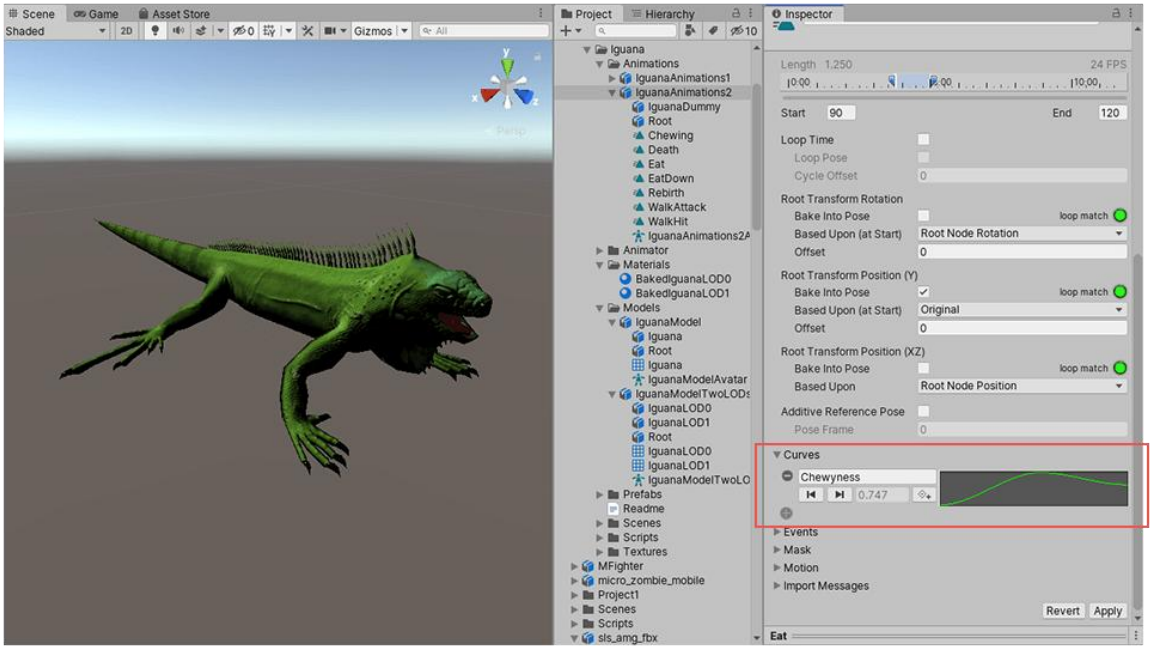
Можете да ги користите овие криви за да додадете дополнителни анимирани податоци на внесениот клип. Можете да ги користите тие податоци за да ги анимирате тајмингот на други ставки врз основа на состојбата на аниматорот. На пример, во игра поставена во ледени услови, дополнителна крива на анимација може да се користи за да се контролира стапката на емисија на системот за честички за да се прикаже здивот на играчот во студениот воздух.

За да додадете крива на внесената анимација, проширете го делот Curves на дното од картичката Анимација и кликнете на иконата плус за да додадете нова крива на тековниот клип за анимација:



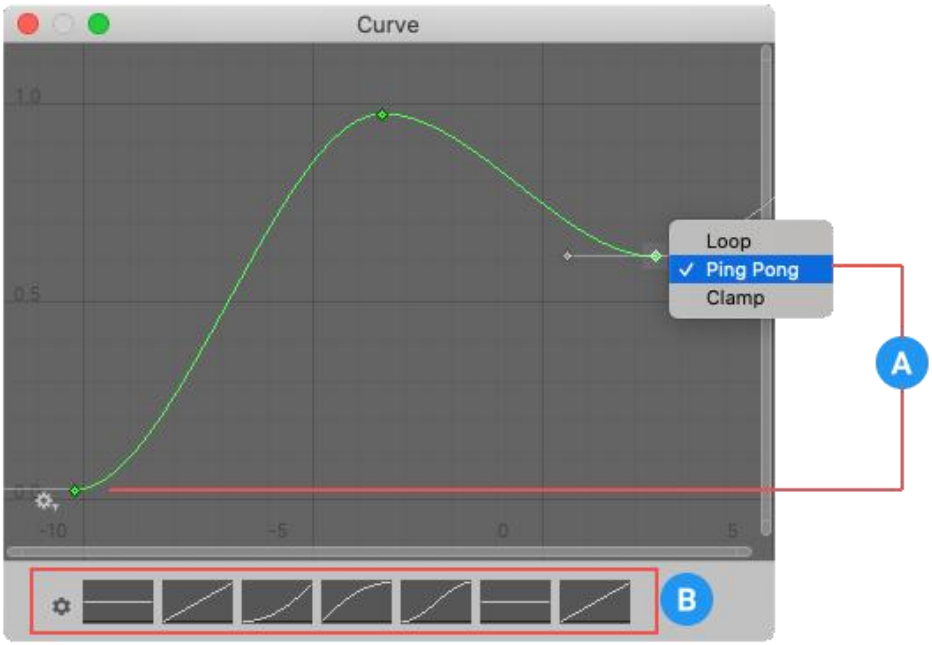
2- 29 Секцијата за проширување на криви на картичката за анимација

Ако вашата внесена датотека со анимација е поделена на повеќе анимациски клипови, секој клип може да има свои сопствени криви.



2- 30 Кривите на внесениот клип за анимација

Х-оската на кривата претставува нормализирано време и секогаш се движи помеѓу 0,0 и 1,0 (што одговара на почетокот и на крајот на клипот за анимација, соодветно, без оглед на неговото времетраење).



2- 31 Уредувач на кривата во Unity

(А) Режим на завиткување

(Б) Поставки за крива

Со двоен клик на кривата на анимација се појавува стандардниот уредувач на кривата, кој можете да го користите за додавање клучеви на кривата. Копчињата се точки по должината на временската линија на кривата каде што има вредност експлицитно поставена од аниматорот наместо само да користи интерполирана вредност. Копчињата се многу корисни за обележување важни точки во должина на временската линија на анимацијата. На пример, со анимација за одење, може да користите копчиња за да ги означите точките каде што левата нога е на земја, потоа двете стапала на земја, десната нога на земја итн. Откако ќе се постават копчињата, можете лесно да се движите помеѓу клучните рамки со притискање на копчињата Previous Key Frame и Next Key Frame. Ова ја поместува вертикалната црвена линија и го прикажува нормализираното време на клучниот кадар. Вредноста што ја внесувате во полето за текст ја движи вредноста на кривата во тоа време.

Криви на анимација и параметри на контролорот на аниматорот

Ако имате крива со исто име како еден од параметрите во контролорот за анимација, тогаш тој параметар ја зема својата вредност од вредноста на кривата во секоја точка од временската линија. На пример, ако упатите повик до GetFloat од скрипта, вратената вредност е еднаква на вредноста на кривата во моментот на повикот. Забележете дека во секој даден момент во времето, може да има повеќе анимациски клипови кои се обидуваат да го постават истиот параметар од истиот контролер. Во тој случај, Unity ги спојува вредностите на кривата од повеќекратните анимациски клипови. Ако анимацијата нема крива за одреден параметар, тогаш Unity се меша со стандардната вредност за тој параметар.

Мешање & Транзиции

Вообичаена задача во анимацијата на играта е да се спојат две или повеќе слични движења. Можеби најпознатиот пример е мешањето на анимации за одење и

трчање според брзината на ликот. Друг пример е лик што се навалува налево или надесно додека се врти за време на трчање.

Важно е да се направи разлика помеѓу транзиции и стебла на мешање (Blend Trees). Додека и двете се користат за создавање мазна анимација, тие се користат за различни видови ситуации.

- Транзициите се користат за непречено преминување од една состојба на анимација во друга во одредено време. Транзициите се наведени како дел од машината за состојба на анимација. Преминот од едно движење на сосема друго движење обично е во ред ако транзицијата е брза.
- Стебла на мешање се користат за да се овозможи непречено мешање на повеќе анимации со инкорпорирање на делови од сите во различен степен. Количината што секое од движењата придонесува за конечниот ефект се контролира со помош на параметар за мешање, кој е само еден од нумеричките параметри за анимација поврзани со контролорот за анимација. За да може мешаното движење да има смисла, движењата што се мешаат мора да бидат од слична природа и време. Стеблата на мешање се посебен тип на состојба во машина за анимација.

Примери за слични движења може да бидат различни анимации за одење и трчање. За да може мешавината да работи добро, движењата во клиповите мора да се одвиваат во истите точки во нормализирано време. На пример, анимациите за одење и трчање може да се усогласат така што моментите на контакт на стапалото со подот се одвиваат во истите точки во нормализирано време (на пример, левата нога удира на 0,0 и десната нога на 0,5). Бидејќи се користи нормализирано време, не е важно дали клиповите се со различна должина.

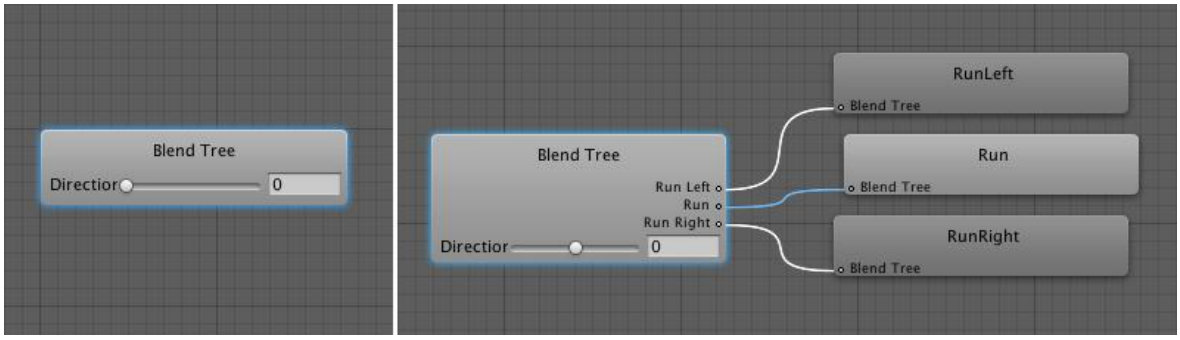
Користење на стебла на мешање

За да започнете да работите со ново стебло на мешање, треба да:

1. Кликнете со десното копче на празниот простор на прозорецот на контролорот на аниматорот.

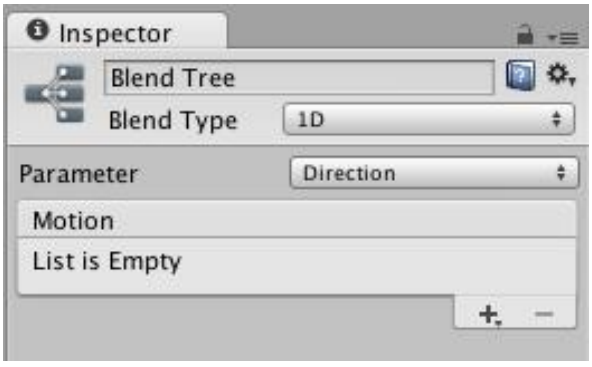
2. Изберете Create State > From New, стебло на мешање (Blend Tree) од контекстното мени што се појавува.
3. Кликнете двапати на стеблото на мешање за да влезете во графиконот на стеблото на мешање.

Прозорецот Аниматор сега прикажува график на целото стебло на мешање, додека инспекторот го прикажува моментално избраниот јазол и неговите непосредни деца.



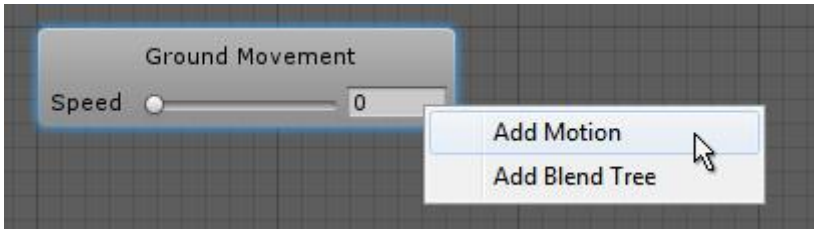
2- 32 Прозорецот Аниматор покажува график на целото стебло на мешање. Лево е стебло на мешање со само коренот јазол за мешање (сè уште не се додадени детски јазли). Десно е стеблото на мешање со коренот јазол за мешање и три анимациски клипови како детски јазли.

За да додадете анимациски клипови во стеблото на мешање, можете да го изберете стеблото на мешање, а потоа кликнете на иконата плус во полето за движење во инспекторот.



2- 33 Јазол за мешање прикажан во инспекторот пред да се додадат какви било движења. Иконата плус се користи за додавање анимациски клипови или деца на стеблото на мешање.

Алтернативно, можете да додадете анимациски клипови или јазли со десен клик на сребло то на мешање и избирање од контекстното мени:

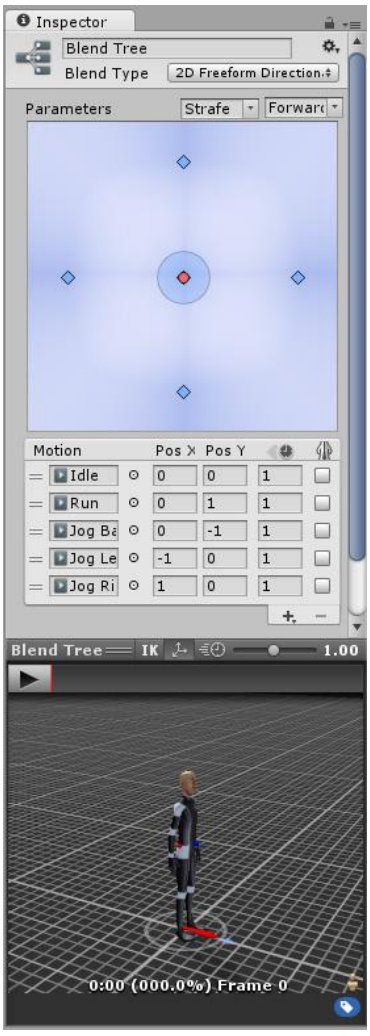


2- 34 Контексното мени кога се кликнува со десното копче на јазолот од стеблото на мешање.

Кога стеблото на мешање е поставен со анимациски клипови и влезни параметри, прозорецот на инспекторот дава графичка визуелизација за тоа како се комбинираат анимациите како што се менува вредноста на параметарот (како што го влечете лизгачот, стрелките од коренот на дрвото го менуваат нивното засенчување за да се прикаже доминантен клип за анимација).

Можете да изберете кој било од јазлите во графикот на стеблото на мешање за да го прегледате во Инспекторот. Ако избраниот јазол е клип за анимација, ќе се прикаже Инспекторот за тој клип за анимација. Поставките ќе бидат само за читање ако анимацијата е внесена од модел. Ако јазолот е јазол за мешање, ќе се прикаже Инспекторот за јазли за мешање.

Можете да изберете или 1D или 2D мешање од менито Тип на мешање; разликите помеѓу двата типа се опишани на нивните сопствени страници во овој дел.



2- 35 2D Стебло на мешање поставено со пет анимациски клипови, кои се прегледуваат во инспекторот

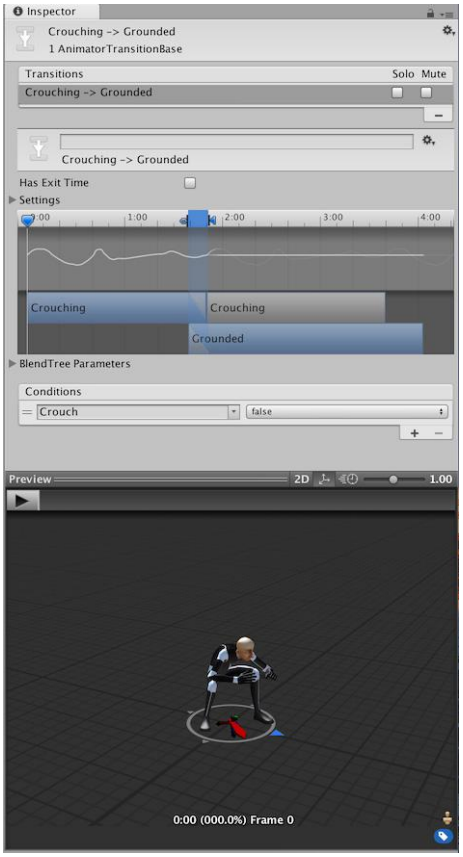
Стебла на мешање и движење на коренот

Мешањето помеѓу анимациите се постапува со помош на линеарна интерполација (т.е., количината на секоја анимација е просек од одделните анимации мерена со параметарот за мешање). Сепак, треба да забележите дека движењето на коренот не се интерполира на ист начин.

Транзиции на анимација

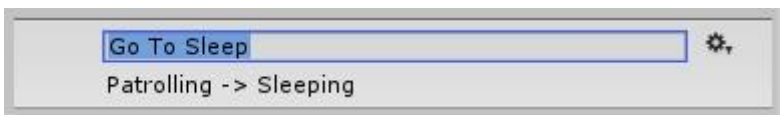
Транзициите на анимацијата и овозможуваат на состојбата на машината да се префрла или да се меша од една состојба на анимација во друга. Транзициите дефинираат не само колку долго треба да трае мешањето помеѓу состојбите, туку и под кои услови тие треба да се активираат. Може да поставите транзиција да се случува само кога одредени услови се вистинити. За да ги поставите овие услови, наведете ги вредностите на параметрите во контролорот за анимација.

На пример, вашиот лик може да има состојба на „патролирање“ и состојба на „спиење“. Може да поставите премин помеѓу патролирање и спиење да се случува само кога вредноста на параметарот „внимание“ ќе падне под одредено ниво.



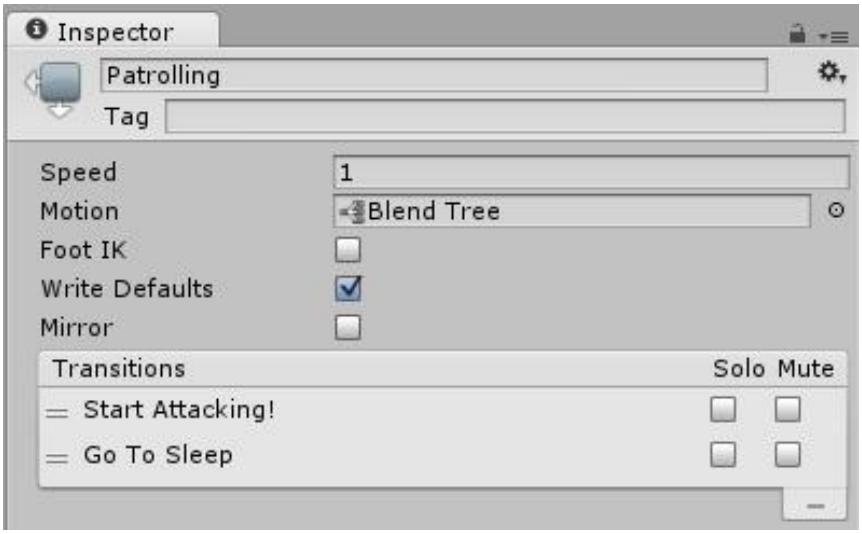
2- 36 Пример за транзиција како што се гледа во инспекторот.

За да им дадете име на транзициите, напишете го во полето како што е прикажано подолу:



2- 37 Именување на транзиции

Прозорецот Инспектор на состојбата ги прикажува транзициите што состојбата ги користи како што е прикажано подолу:

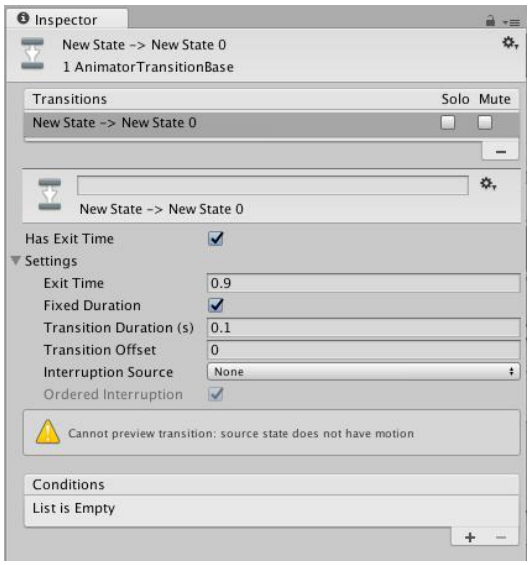


2- 38 Транзициите што состојбата ги користи

Може да има само една транзиција активна во даден момент. Меѓутоа, моментално активната транзиција може да биде прекината со друга транзиција ако сте ги конфигурирале поставките да го дозволат тоа (видете Прекин на транзиција подолу).

Карактеристики на транзиција

За да ги видите својствата за транзиција, кликнете на линијата за преод што поврзува две состојби во прозорецот Аниматор. Својствата се појавуваат во прозорецот Инспектор.



2- 39 Поглед на својствата на транзицијата

Својство	Функција
Има време на излез	Времето на излегување е специјална транзиција што не се потпира на параметар. Наместо тоа, се потпира на нормализираното време на состојбаат. Проверка за да се изврши транзицијата во одреденото време наведено во времето на излез.
Поставки	Преклопено мени кое содржи детални поставки за транзиција како подолу.
Време на излез	Ако времето за излез е означено, оваа вредност го претставува точното време во кое транзицијата може да стапи на сила. Ова е претставено во нормализирано време (на пример, време на излегување од 0,75 значи дека на првиот кадар каде што е пуштена 75% од анимацијата, условот време на излез е точен). На следната рамка, состојбата е неточна. За анимации со јамка, транзициите со излезни времиња помали од 1 се оценуваат во секоја јамка, така што можете да го користите ова за да ја темпирате вашата транзиција со соодветно време во анимацијата на секоја јамка. Транзициите со време на излез поголем од 1 се оценуваат само еднаш, така што може да се користат за излез во одредено време по фиксен број на циклуси. На пример, транзицијата со време на излез од 3,5 се оценува еднаш, по три и пол јамки.

Фиксно времетраење	Ако полето на фиксното времетраење е означено, времето на транзиција се толкува во секунди. Ако полето не е означено, времето на транзиција се толкува како дел од нормализираното време на изворната состојба.
Времетраење на транзицијата	Времетраењето на транзицијата, во нормализирано време или секунди во зависност од режимот на фиксното времетраење, во однос на времетраењето на моменталната состојба. Ова е визуелизирано во графикот на транзиција како дел помеѓу двата сини маркери.
Офсет на транзиција	Поместување на времето за започнување на играње во дестинација во која се префрла. На пример, вредноста од 0,5 значи дека целната состојба почнува да игра на 50% од патот низ сопствената временска линија.
Извор на прекин	Користете го ова за да ги контролирате околностите под кои оваа транзиција може да биде прекината (видете Прекин на транзиција подолу).
Нареден прекин	Определува дали тековната транзиција може да биде прекината со други транзиции независно од нивниот редослед (види Прекин на транзиција подолу).
Услови	Транзицијата може да има единствен услов, повеќе услови или воопшто да нема услови. Ако вашата транзиција нема услови, Уредникот на Unity го разгледува само времето на излез, а транзицијата се случува кога ќе се достигне времето на излез. Ако вашата транзиција има еден или повеќе услови, сите услови мора да бидат исполнети пред да се активира транзицијата. Условот се состои од: - параметар на настан (вредноста што се разгледува во условот). - условен прирок (ако е потребно, на пример, „помалку од“ или „повеќе од“ за плови). - вредност на параметарот (ако е потребно). Ако го имате избрано времето на излез за транзицијата и имате еден или повеќе услови, имајте во предвид дека Уредникот на Unity разгледува дали условите се вистинити по времето на излез. Ова ви овозможува да се осигурате дека вашата транзиција се случува за време на одреден дел од анимацијата.

Прекин на транзиција

Користете ги својствата Извор на прекин и Нареден прекин за да контролирате како вашата транзиција може да биде прекината. Редоследот на прекин работи, концептуално, како да транзициите се редат и потоа се анализираат за валидна транзиција од првата вметната транзиција до последната.

Прекин на својство на извор

Транзициите во AnyState секогаш се додаваат прво во редот, а потоа другите транзиции се редат во зависност од вредноста на Изворот на прекин:

Вредност	Функција
Нема	Не додава повеќе транзиции.
Моментална состојба	Поставување во редица на транзициите од моменталната состојба.
Следна состојба	Поставување во редица на транзициите од следната состојба.
Тековна состојба, потоа Следна состојба	Поставување во редица на транзициите од моменталната состојба, а потоа оние од следната состојба.
Следна состојба потоа Тековна состојба	Поставување во редица на транзициите од следната состојба, па оние од моменталната состојба.

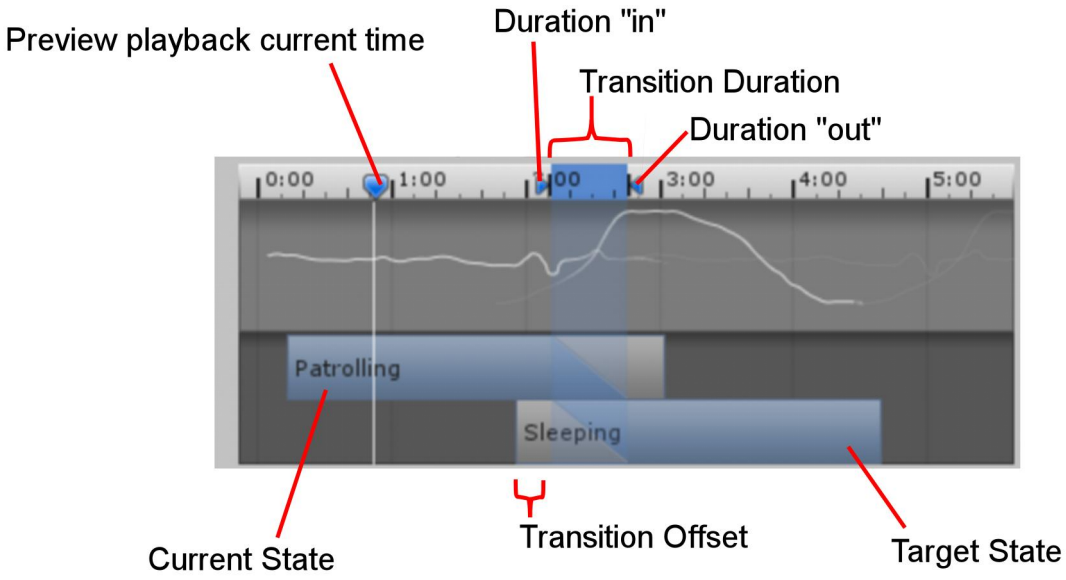
Својство на нареден прекин

Својството на нарачан прекин го менува начинот на анализирање на редот. Во зависност од неговата вредност, парсирањето на редот завршува во различен момент како што е наведено подолу. Само транзицијата на AnyState може сама по себе да се прекине.

Вредност	Завршува кога
означено	Кога е пронајдена валидна транзиција или тековната транзиција.
неозначено	Пронајдена е валидна транзиција.

График на транзиција

За рачно прилагодување на поставките наведени погоре, можете или да внесете броеви директно во полињата или да го користите графикот за транзиција. Графикот на транзиција ги модифицира горните вредности кога се манипулира со визуелните елементи.



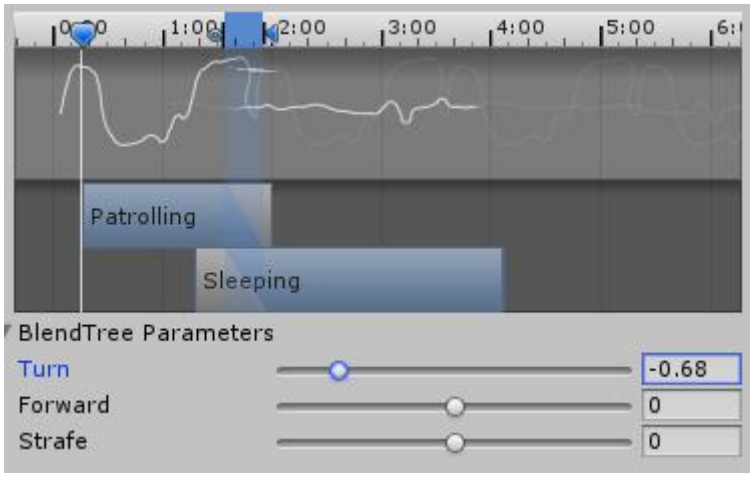
2- 40 Поставките за транзиција и графикот како што е прикажано во Инспекторот

Променете ги својствата на транзицијата во приказот на графиконот користејќи ги следните насоки:

- Повлечете го маркерот Времетраење „надвор“ за да го промените времетраењето на транзицијата.
- Повлечете го маркерот Времетраење “внатре” за да го промените времетраењето на транзицијата и времето на излез.
- Повлечете ја целната состојба за да го прилагодите преодното поместување.
- Повлечете го маркерот за репродукција за преглед за да се пробие низ мешавината на анимација во прозорецот за преглед на дното на Инспекторот.

Транзиции помеѓу состојбите на стеблото на мешање

Ако тековната или следната состојба што припаѓа на оваа транзиција е состојба на стеблото на мешање, параметрите на стеблото на мешање се појавуваат во Инспекторот. Прилагодете ги овие вредности за да прегледате како ќе изгледа вашата транзиција со вредностите на стеблото на мешање поставени на различни конфигурации. Ако вашето стебло на мешање содржи клипови со различна должина, треба да тестирате како изгледа вашата транзиција кога ги прикажувате краткиот и долгиот клип. Прилагодувањето на овие вредности не влијае на тоа како се однесува транзицијата при извршување; тие се само за да ви помогнат да прегледате како може да изгледа транзицијата во различни ситуации.



2- 41 Контролите за преглед на параметарот стебло на мешање, видливи кога вашата тековна или следната состојба е состојба на стебло на мешање.

2.1.5 Скриптирање и кодирање во Unity

Овој пододдел се фокусира на методологијата на скриптирање и кодирање во рамките на Unity погонот. Опфаќа теми како што се јазик за скриптирање C#, работа со компоненти и GameObjects, имплементирање на логиката на играта и користење на API на Unity за различни функционалности, вообичаени шеми на кодирање, техники за дебагирање и најдобри практики за ефикасен и одржуван код во проектите на Unity. [16, 35, 50, 52]

Скриптирањето е суштинска состојка во сите апликации што ги правите во Unity. На повеќето апликации им се потребни скрипти за да одговорат на влезот од играчот и да организираат настани во играта да се случат кога треба. Покрај тоа, скриптите може да се користат за создавање графички ефекти, контрола на физичкото однесување на предметите или дури и имплементација на сопствен систем за вештачка интелигенција за ликовите во играта.

Unity API

Unity API (Application Programming Interface) е збирка на класи, методи и структури на податоци обезбедени од Unity Technologies за да им овозможат на програмерите да комуницираат со и да го контролираат погонот за игри Unity. API делува како мост помеѓу кодот на играта (напишан во C# или UnityScript) и основниот погон, овозможувајќи им на програмерите да манипулираат со предметите на играта, средствата, физиката, аудиото, корисничките интерфејси и многу повеќе.

Unity API е обемна и добро документирана, обезбедувајќи широк опсег на функционалности за креирање, менување и управување со различни аспекти на играта. Некои клучни области опфатени со Unity API вклучуваат:

- GameObjects: манипулирање и комуницирање со објектите во играта, како што се создавање, движење, скалирање и ротирање.
- Компоненти: пристап и контрола на различни компоненти прикачени на GameObjects, како што се судири, крути тела, скрипти и системи со честички.

- Физика: Спроведување физички интеракции користејќи го вградениот физички погон, примена на сили, откривање на судири и справување со динамиката на крутото тело.
- Влез: Снимање на влез од играчот од различни уреди, како што се тастатури, глувци, екрани на допир и контролери.
- Графика: Управување со сенките, текстурите, материјалите и поставките за рендерирање за да го контролирате визуелниот изглед на играта.
- Аудио: Ракување со аудио извори, слушатели, и просторен звук за да создадете извонредни аудио- искуства.
- UI: Дизајнирање и управување со кориснички интерфејси користејќи го системот за интерфејс на Unity, вклучувајќи копчиња, текстуални елементи и панели.
- Управување со средства: вчитување, растоварување и управување со средствата на играта како модели, текстури, аудио клипови и многу повеќе.
- Управување со сцена: Контролирање на текот на играта со вчитување, растоварување и преминување помеѓу различни сцени.
- Вмрежување: имплементирање функционалност за повеќе играчи преку Unity Networking API.

Програмерите можат да пристапат до Unity API од нивните скрипти и да го прилагодат однесувањето на нивната игра според нивните специфични потреби. Со овој моќен интерфејс, Unity ги овластува развивачите на игри да изградат широк спектар на интерактивни и извонредни искуства на различни платформи и уреди.

C#

C# (се изговара Си-Шарп), инспириран од многу карактеристики на F#, е програмски јазик кој првпат се појави во јавноста во 2000 година. Тој е јазик за општа намена дизајниран за развивање апликации на Windows и бара .NET Framework да работи. Јазикот може да се користи за програмирање речиси сè. Но, тој е добар за развој на апликации и игри за Windows. Некои веб-програмери, исто така, избираат C# за да направат веб-апликација, а исто така е популарен и за развој со мобилни телефони.



2- 42 Лого на C#

C# е лесен за учење, релативно лесен за читање, но поради неговата флексибилност, тешко е да се совлада сето она што C# може да го понуди. C# е сложен јазик, за целосно совладување може да биде потребно повеќе време од поедноставен јазик како што се Python или HTML, корисниците треба да научат значителна количина на код за да создадат напредна програма. Но, вреди да се посвети времето на корисниците за да го научат тоа. C# е вештина на што ја бараат многу технолошки компании. На пазарот, секојдневно и секаде има апликации на кои им требаат програмери со добро знаење и искуство во C#. Затоа, луѓето кои учат C# имаат одлични можности за работа. Го избрав C# поради неговата флексибилност. Читателите можат да научат C# не само за да прават игри, туку и да имаат подобра можност во нивното патување да станат развивач на кој било аспект од ИТ светот.

Компоненти

Компонентите се функционални делови на секој GameObject. Компонентите содржат својства кои можете да ги уредите за да го дефинирате однесувањето на објектот.

За да видите листа на компоненти прикачени на Game објектот во прозорецот Инспектор, изберете го Game објектот или во прозорецот Хиерархија или во сцената.

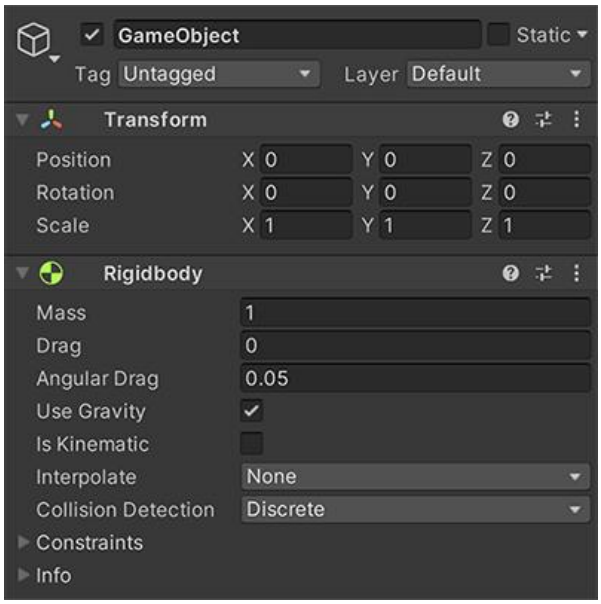
Можете да прикачите многу компоненти на еден Game Object, но секој Game Object мора да има една и само една Transform компонента. Ова е затоа што компонентата Transform ја диктира локацијата, ротацијата и размерот на објектот. За да креирате празен Game Object, изберете Game Object > Create Empty. Кога ќе го изберете новиот Game Object, инспекторот ја прикажува компонентата Transform со стандардни вредности.

Користење на компоненти

Можете да користите различни компоненти за менување или додавање функции на вашите објекти. Можете да го користите прозорецот Инспектор за да ги промените својствата на која било компонента или можете да користите скрипти.

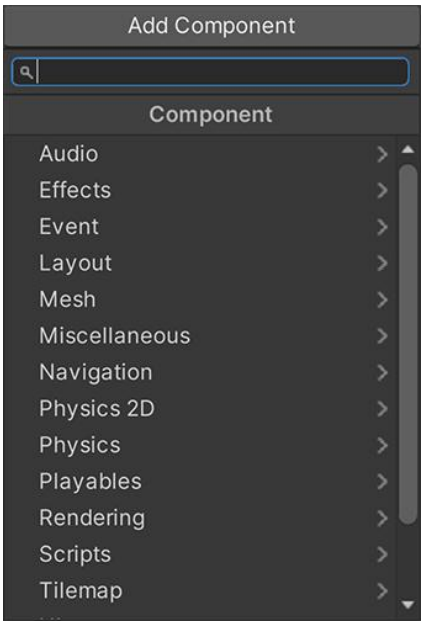
Додавање на компоненти

Можете да додадете компоненти на избраниот Game Object преку менито Component. За да додадете компонента Rigidbody, изберете Game Object и изберете Component > Physics > Rigidbody од менито. Инспекторот ги прикажува својствата на Rigidbody. Ако притиснете Play додека го имате избрано празен GameObject, позицијата Y на трансформацијата на Game Object се намалува. Тоа е затоа што физичкиот систем во Unity предизвикува GameObject да падне под гравитација.



2- 43 Game Object со прикачена компонента Rigidbody

Можете исто така да додавате компоненти во прелистувачот на компоненти. За да го отворите прелистувачот на компоненти, изберете Додај компонента во Инспекторот.



2- 44 Прелистувач на Компоненти

Можете да се движите низ компонентите по категорија во прелистувачот или да го користите полето за пребарување за да ги лоцирате компонентите по име.

Можете да прикачите кој било број или комбинација на компоненти на еден GameObject. Некои компоненти најдобро функционираат во комбинација со други. На пример, Rigidbody работи со Collider.

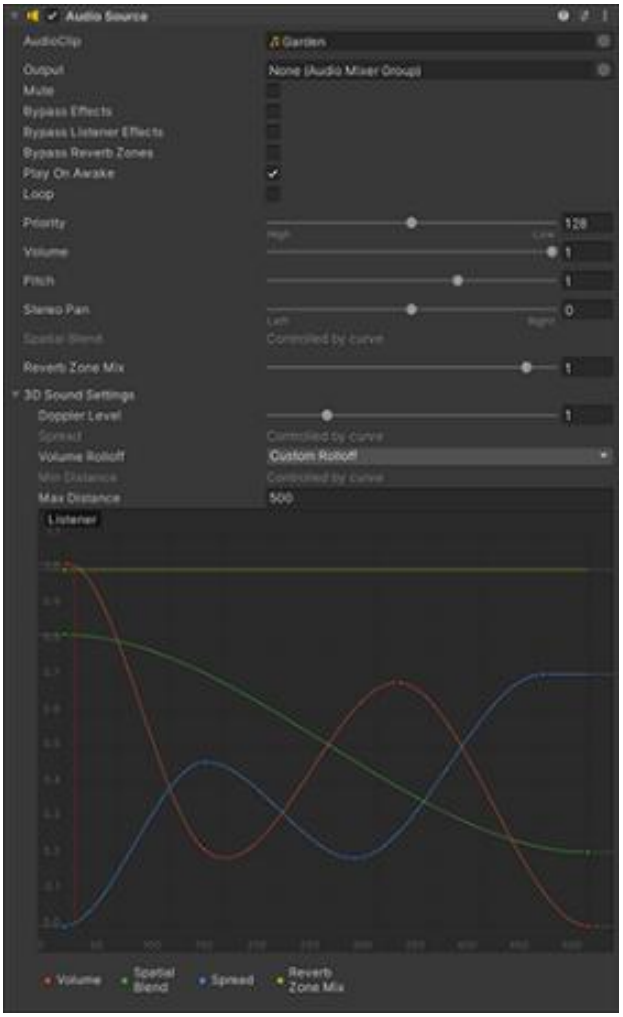
Уредување на компонентите

Кога прикачувате компонента на Game Object, својствата на компонентата содржат стандардни вредности. Можете да ги уредувате овие вредности во Уредувачот додека градите игра или во скриптите кога ја извршувате играта. Постојат два главни типа на својства: вредности и референци.

Можете да ги уредувате својствата на вредноста во Инспекторот. Постојат различни типови на вредности, вклучувајќи текст, преклопувања и паѓачки мениа.

За референтни својства, можете да повлечете датотека од приказот на проектот во својството или да го користите избирачот на објект (икона круг) на својството. Референтните својства може да упатуваат на други видови компоненти, Game Objects или средства.

Сликата 2-45 покажува Game Object со компонента Audio Source. Вредностите на Аудио изворот во Инспекторот покажуваат како можете да ги прилагодите аспектите на компонентата за да одговараат на вашиот проект.



2- 45 Аудио изворот со звучен ефект е референциран правилно во аудио клипот

Тестирање својства

Во режимот за репродукција, можете да ги промените својствата на компонентите во Инспекторот. Ова ви овозможува да видите како различните вредности влијаат на играта. На пример, можете да експериментирате со различни висини на скокање. Ако креирате својство Jump Height во скрипта, можете да влезете во режим на репродукција, да ја промените вредноста и да го притиснете копчето за скок за да видите што се случува. Потоа, без да излезете од режимот на репродукција, можете повторно да го промените и да ги видите резултатите за неколку секунди. Кога ќе излезете од „Режимот за репродукција“, својствата се враќаат на нивните предходни вредности. Со овој работен тек, можете да експериментирате, приспособите и да ја усовршите вашата игра за помалку време.

Логика на играта

Логиката на играта е столбот на секое интерактивно искуство во Unity. Опфаќа збир на правила, механики и системи кои диктираат како работи видео играта и како реагира на внесувањето на играчите. Логиката на играта е одговорна за управување со движењата на карактерите, интеракциите со предметите, условите за победа/пораз, бодување и многу повеќе. Како програмери, создаваме скрипти и код за да ја имплементираме оваа логика, дефинирајќи како се однесува светот на играта и како различните елементи во него комуницираат едни со други и со играчот. Со внимателна изработка на логиката на играта, може да се создадат привлечни и извонредни искуства кои ги предизвикуваат и забавуваат играчите, а истовремено обезбедува кохезивен и пријатен тек на игра. Ефективната употреба на скриптирање и кодирање во Unity ни овозможува да им вдахнеме живот во нашите игри, правејќи ги динамични, одговорни и на крајот исполнувачки за играчите да ги истражуваат и освојуваат.

Вообичаени модели на кодирање

Моделите за кодирање се стандардизирани пристапи за решавање на специфични програмски проблеми или имплементација на одредени функционалности. Тие помагаат да се подобри читливоста на кодот, одржливоста и повторното користење. Еве неколку вообичаени обрасци за кодирање кои често се користат во развојот на софтвер:

- Factory Pattern: Обезбедува интерфејс за креирање објекти, но им овозможува на подкласите да го менуваат типот на објекти што ќе се креираат. Корисно за создавање објекти без изложување на логиката на инстанцата.
- Шема на стратегија: Дефинира фамилија на алгоритми, го опфаќа секој од нив и ги прави заменливи. Оваа шема му овозможува на клиентскиот код да го избере соодветниот алгоритам без да ја менува неговата структура.
- Шема на декоратор: Дозволува однесувањето да се додаде на поединечни објекти, статички или динамички, без да влијае на однесувањето на другите објекти од истата класа.
- Builder Pattern: Одвојува конструкција на комплексен објект од неговата репрезентација, дозволувајќи му на истиот процес на градба да создаде различни претстави.
- Шема на Итератор: Обезбедува начин за пристап до елементите на збирката последователно без изложување на нејзината основна репрезентација.
- Шаблонски метод: Ја дефинира структурата на алгоритамот во методот, но им овозможува на поткласите да отфрлаат одредени чекори на алгоритмот без да ја менуваат неговата структура.

Овие обрасци не се ексклузивни за Unity или кој било специфичен јазик; тие можат да се имплементираат во различни програмски средини. Разбирањето и користењето на овие шеми на кодирање може значително да ја подобри организацијата и ефикасноста на вашата база на кодови. Сепак, важно е да ги користите разумно и да ги приспособите по потреба за да одговараат на специфичните барања на вашиот проект.

Техники за дебагирање

Дебагирање е процес на идентификување и решавање на проблеми или грешки во софтверот. Тоа е суштинска вештина за програмерите да обезбедат нивниот код да работи како што се очекува. Еве неколку вообичаени техники за дебагирање што може да се користат за отстранување и поправка на проблеми:

- Печатење на изјави: Вметнувањето на изјави во вашиот код за прикажување на променливи вредности, пораки или контролни точки за време на траењето

може да ви помогне да го разберете текот на вашата програма и да идентификувате каде може да се појават проблеми.

- Алатки за дебагирање: Современите интегрирани развојни околини (IDE) нудат моќни алатки за дебагирање. Овие алатки ви дозволуваат да поставите точки на прекин, да ги прегледате вредностите на променливите, да чекорите низ извршувањето на кодот и да ги анализирате куповите на повици. Разбирањето и користењето на овие карактеристики може значително да го забрза процесот на дебагирање.
- Евиденција: Пишувањето детални пораки од дневникот во датотека или конзола може да обезбеди вредни информации за извршувањето на програмата, дури и кога не можете да пристапите до околината за отстранување грешки.
- Тестирање на единицата: Пишувањето сеопфатни тестови на единицата за вашиот код може да помогне рано да се фатат грешките и да се обезбеди безбедносна мрежа кога се прават промени во базата на кодови.
- Преглед на кодови: Друг програмер да го прегледа вашиот код може да открие потенцијални проблеми и да понуди нови перспективи за тоа како да го подобрите квалитетот на кодот.
- Дебагирање на гумена патка: Објаснувањето на вашиот код или проблемот со кој се соочувате на некој друг, може да ви помогне да го видите проблемот од поинаков агол и да ве наведе да го пронајдете решението.
- Бинарен метод на пребарување: Кога се занимавате со голем или сложен код, поделете го проблемот на половина и тестирајте ја средната точка. Во зависност од резултатот, фокусирајте се на првата или втората половина и повторете ја постапката додека не се изолира проблемот.
- Контрола на верзијата: Користењето системи за контрола на верзии (на пр., Git) ви овозможува да ги следите промените во вашата база на кодови, да се враќате на претходните состојби и ефикасно да соработувате со другите. Ова може да биде неизмерно корисно за идентификување кога е воведен проблем.
- Испитајте ја документацијата: Понекогаш, проблемите се јавуваат поради неточни претпоставки за тоа како функционираат одредени функции или библиотеки. Двојната проверка на документацијата и примерите може да ви помогнат да забележите такви проблеми.

- Тестирање со изолација: Кога ќе наидете на грешка, обидете се да создадете минимален пример или да го изолирате проблематичниот код во помал контекст. Ова може да ви помогне да ја разберете основната причина и побрзо да најдете решение.
- Далечинско отстранување грешки: за проблеми кои се јавуваат само во одредени средини (на пр., на оддалечен сервер или одредена платформа), користете далечински алатки за отстранување грешки или анализа на дневници за да го истражите проблемот.

Запомнете, дебагирањето е вештина што се подобрува со вежбање. Трпението и упорноста се од суштинско значење кога се решаваат сложени проблеми. Исто така, не заборавајте да ги документирате вашите наоди и решенија за да си помогнете себеси и на другите во иднина.

Ефикасен код

Пишувањето ефикасен код е клучно за подобрување на перформансите, намалување на искористеноста на ресурсите и создавање на одговорни и скалабилни апликации. Еве неколку совети за пишување ефикасен код:

- Избор на алгоритам: Изберете ги вистинските алгоритми и структури на податоци за вашиот конкретен проблем. Размислете за нивната временска и просторна сложеност и како тие се размеруваат со влезната големина.
- Намалете го вишокот: Избегнувајте дуплирање на код со користење на функции, класи и модули за да се инкапсулираат заедничката функционалност. Кодот за повеќекратна употреба води до полесно одржување и помала база на кодови.
- Минимизирајте ги јамките и вгнездените јамки: Јамките може да бидат скапи, особено вгнездените јамки. Барајте можности за оптимизирање и намалување на бројот на повторувања.
- Користете вградени функции и библиотеки: користете ги вградените функции и стандардните библиотеки секогаш кога е можно, бидејќи тие обично се оптимизирани за перформанси и доверливост.

- Мрзливо вчитување и одложено извршување: Вчитувајте податоци или иницијализирајте објекти само кога е потребно. Овој пристап може да помогне да се намали употребата на меморијата и да се подобри времето на стартување.
- Избегнувајте глобални променливи: минимизирајте ја употребата на глобалните променливи, бидејќи тие може да доведат до неочекувани несакани ефекти и да го отежнат расудувањето за кодот.
- Управување со меморијата: Внимавајте на распределбата и распоредувањето на меморијата. Избегнувајте непотребно создавање на објекти и претпочитајте здружување на објекти каде што е соодветно.
- Сериска обработка: Ако работите со голем број слични операции, обидете се да ги здружите заедно за да ги намалите трошоците.
- Користете го забрзувањето на графичкиот процесор: во одредени сценарија, исфрлете ги пресметковно интензивните задачи на графичкиот процесор за побрза обработка.
- Профилирање: Идентификувајте ги тесните грла и проблемите со перформансите со користење на алатки за профилирање за мерење на времето на извршување на различни делови од вашиот код.
- Оптимизирајте ги I/O операциите: Минимизирајте го читањето и запишувањето на дискот, особено за време на критичните делови од апликацијата.
- Асинхроно програмирање: Користете техники за асинхроно програмирање за да ја подобрите одзивноста и подобро да ги искористите системските ресурси.
- Избегнувајте предвремена оптимизација: прво фокусирајте се на пишување чист и оддржлив код. Оптимизирајте ги само критичните делови за перформанси кога е потребно, врз основа на резултатите од профилирањето.
- Резултати од кешот: Кеширајте ги резултатите од скапите пресметки или барањата во базата на податоци за да избегнете непотребна работа.
- Преглед на код: нека го прегледаат вашиот код од колегите за да добиете увид и предлози за потенцијални подобрувања.
- Чувајте ги апстракциите едноставни: Избегнувајте непотребна сложеност во вашите апстракции. Одржувајте го вашиот дизајн и код директен и лесен за разбирање.

- Тестирање: Темелно тестирајте го вашиот код за да се уверите во исправност и рано да ги идентификувате проблемите со перформансите.

Запомнете дека ефикасноста е компромис помеѓу различни фактори како што се читливоста на кодот, одржливоста и перформансите. Секогаш стремете се кон рамнотежа што ги задоволува барањата на вашиот проект. Оптимизирањето на секој дел од кодот не е секогаш неопходно, но насочувањето на критичните области може да доведе до значителни севкупни подобрувања.

III. АНАЛИЗА НА ИСТРАЖУВАЊЕТО, МЕТОДОЛОГИЈА И ДИСКУСИЈА

3 ИСТРАЖУВАЊЕТО И ИНТЕГРИРАЊЕТО НА ПРИНЦИПИТЕ ЗА РАЗВОЈ НА ИГРИ СО UNITY

Овој дел ја истражува практичната примена на дел од принципите за развој на игри дискутирани во Дел I и II во рамките на околината за развој на игри.

3.1 Методологија на преглед на литература

Овој поддел ја објаснува методологијата користена за спроведување на прегледот на литературата. Ги вклучува критериумите за избор на извори, процесот на пребарување и анализата на литературата. Се дискутира за прашањата или целите на истражувањето кои го водат прегледот на литературата, базите на податоци или пребаруваните извори и користените клучни зборови или термини за пребарување. Исто така, ги наведува критериумите за вклучување и исклучување за избор на релевантна литература и дава образложение за избраниот пристап.

Критериуми за избор на извори

Прегледот на литературата за овој магистерски труд на тема „Развој на игри со Unity: дводелно патување во создавање и имплементација“ употреби специфични критериуми за избор за да се идентификуваат релевантни и сигурни извори. Обемот на трудот опфати неколку теми во теоријата за развој на игри, вклучувајќи теорија на дизајн на игри, психологија на играчите во игрите, раскажување приказни, техники во игрите, анализа на механиката на игрите и шеми на дизајн во развој на игри. Критериумите за избор имаа за цел да обезбедат вклучување на сеопфатни и разновидни перспективи додека ја одржуваа релевантноста за целите на тезата.

Различни видови извори беа разгледани за да се обезбеди сеопфатно разбирање на избраните теми. Истражувањето вклучи академски списанија, книги, конференциски трудови и реномирани индустриски извештаи. Вклучувањето на

различни извори овозможи добро заокружено истражување на теоретските аспекти на развојот на играта и обезбеди увид и од академски и од индустриски експерти.

За да се осигури дека прегледот на литературата е ажуриран и избалансиран, критериумите за избор не им дадоа приоритет на само неодамнешните публикации, туку ги земаа и во предвид вредните придонеси од постарите дела. Овој пристап ја препозна основната природа на некои теории и принципи на дизајн во развојот на играта, кои можеби биле воспоставени во претходните извори. Како такви, и неодамнешните и постарите дела беа вклучени за да се долови еволуцијата и континуитетот на идеите во оваа област.

Критериумите за избор на извори имаа за цел да создадат цврста и сеопфатна основа за преглед на литературата, овозможувајќи детално истражување на теоретските аспекти на развојот на играта. Со инкорпорирање на разновиден опсег на реномирани извори, прегледот на литературата се стреми да прикаже сеопфатно разбирање на темите што се истражуваат во теоретскиот дел на овој магистерски труд.

Процес на пребарување

Прегледот на литературата за овој магистерски труд, вклучуваше темелен и систематски процес на пребарување за да се идентификуваат релевантните извори. Примарните бази на податоци што се користат за пристап до академски трудови и публикации беа „ResearchGate“ и други реномирани академски складишта. Дополнително, беа истражени различни пребарувачи и онлајн библиотеки за да се лоцираат дополнителни извори кои одговараат на темите што се истражуваат. Имено, се разгледуваше широк опсег на извори за да се осигура дека прегледот на литературата е сеопфатен и добро заокружен.

Специфичните клучни зборови или термини за пребарување користени за време на процесот на пребарување беа насловите на темите идентификувани во теоретскиот дел на трудот.

За да се задржи релевантноста и да се остане актуелен со најновите случувања во областа на развојот на игри, процесот на пребарување се фокусираше на

неодамнешните публикации. Беа направени напори да се даде приоритет на изворите објавени во последните неколку години, сметајќи дека тие повеќе ги одразуваат најсовремените знаења и сознанија.

Анализа на литературата

Анализата на собраната литература за овој магистерски труд, вклучуваше процес на синтетизирање и евалуација на информациите за да се извлечат значајни согледувања. Целта беше да се изгради кохерентно и сеопфатно разбирање на теоретските аспекти на развојот на играта.

За да се синтетизира и анализира литературата, користен е рачен пристап. Релевантните параграфи кои содржат еден или повеќе клучни зборови поврзани со темите на трудот беа идентификувани и внимателно разгледани. Овие параграфи потоа беа комбинирани за да се создаде кохезивен и информативен наратив кој се однесува на прашањата и целите на истражувањето. Процесот на синтеза обезбеди дека клучните концепти и идеи од различни извори се интегрирани во унифицирана рамка.

Оценувањето на кредибилитетот и релевантноста на изворите беше направено со разгледување на нивниот статус на објавување и угледот на авторите и издавачите. Академските книги и трудови беа претпочитани поради нивната рецензирана природа, обезбедувајќи повисоко ниво на кредибилитет. Дополнително, приоритет беа дадени на реномирани издавачи и познати автори, бидејќи нивните придонеси обично се одржуваат според ригорозни стандарди на истражување и преглед.

Во текот на анализата на литературата, немаше значајни предизвици или ограничувања. Процесот беше рационализиран со внимателен избор на веродостојни извори и сеопфатна природа на прегледот на литературата. Синтезата на информациите беше поефикасна со рачно испитување и организирање на релевантната содржина, обезбедувајќи нејзина кохерентност со прашањата и целите на истражувањето.

3.2 Примена на теоријата за дизајн на игри во Unity

Кога развивате игри во Unity, принципите на теоријата за дизајн на игри може ефективно да се применат за да се создадат привлечни и извонредни искуства. Интерфејсот на Unity обезбедува моќна и прифатлива платформа за имплементација на овие принципи. Интерфејсот на Unity се состои од неколку клучни елементи, вклучувајќи го и приказот на сцената, кој им овозможува на програмерите визуелно да дизајнираат и манипулираат со сцените на играта, и приказот на играта, кој обезбедува преглед на играта во реално време како ќе им изгледа на играчите. Инспекторот дозволува проверка и модификација на GameObject-ите и компонентите, додека Хиерархијата обезбедува хиерархиска репрезентација на сите објекти во тековната сцена. Прозорецот на проектот, од друга страна, управува со средствата на проектот, а Конзолата обезбедува дневник на пораки генерирани за време на траењето, помагајќи во дебагирање и оптимизација.

Еден од основните аспекти на теоријата за дизајн на игри применета во Unity е имплементацијата на механиката на играта. Со искористување на способностите за скриптирање на Unity и Unity API, програмерите можат да создадат интерактивни системи кои управуваат со активностите, предизвиците и наградите на играчите, придонесувајќи за целокупното искуство на игра. Дополнително, архитектурата заснована на компоненти на Unity овозможува модуларен дизајн на GameObjects, што го олеснува создавањето на механика за повеќекратна и скалабилна употреба.

Дизајнот на ниво е уште една клучна област каде што теоријата за дизајн на игри наоѓа примена во Unity. Програмерите можат да креираат и организираат сцени користејќи шаблони, рационализирајќи го процесот на создавање ниво. Со поставување камери и додавање на осветлување и сенки, тие можат да создадат визуелно привлечни и атмосферски средини што ја подобруваат естетиката и расположението на играта. Поддршката на Unity за осветлување во реално време и печено осветлување, како и глобалното осветлување, гарантира дека светот на играта се чувствува динамичен и извонреден.

Покрај тоа, анимациите играат витална улога во оживување на игрите. Unity нуди робусен систем за анимација кој им овозможува на програмерите лесно да имплементираат анимации. Тие можат да користат криви за анимација за да го

контролираат времето и интерполацијата на анимациите, обезбедувајќи мазни и природни движења. Стебла на мешање овозможуваат мешање на повеќе анимации, обезбедувајќи беспрекорна транзиција помеѓу различни дејства и состојби.

Во Unity, управувањето со средства е од суштинско значење за ефикасен развој на играта. Програмерите можат да внесуваат и организираат средства, приспособувајќи ги поставките за внес на средства за да ги оптимизираат перформансите и квалитетот. Тие можат да се справат со сложени средства, па дури и да создаваат сопствени средства прилагодени на нивните специфични потреби. Со реекспортирање на пакети на средства, тие можат да споделуваат и повторно да ги користат средствата низ проектите. Употребата на ефективни стратегии за именување обезбедува добро структуриран и одржлив проект.

Конечно, скриптирањето и кодирањето го формираат столбот на развојот на играта во Unity. Користејќи C# и Unity API, програмерите можат да имплементираат логика на играта, да создаваат сопствени компоненти и да се справат со внесувањето на корисникот за да создадат динамични и одговорни искуства за играње. Разбирањето на вообичаените шеми на кодирање и користењето ефикасни практики за кодирање се клучни за оптимизирање на перформансите на играта и обезбедување непречено искуство со играњето.

3.3 Инкорпорирање на психологијата на играчите во игрите

Вклучувањето на психологијата на играчите во игрите е критичен аспект на дизајнот на играта, фокусирајќи се на беспрекорната интеграција на концептите за психологија на играчите во текот на процесот на развој. Со разбирање на мотивацијата, преференциите и однесувањето на играчите, дизајнерите на игри можат да создадат волшебни и извонредни искуства кои се грижат за различни типови играчи и нивните уникатни мотивации за играње. Овај труд навлегува во повеќе димензии на психологијата на играчите, вклучувајќи мотивација, типови на играчи, задоволство на играчите, Доживувањето и влијанието на емоциите врз играта. Исто така, ги истражува техниките на раскажување приказни, нагласувајќи ја структурата на раскажувањето, развојот на ликови, градењето на светот и интерактивното раскажување приказни, кои придонесуваат за подобрување на ангажираноста на играчите. Дополнително, се анализира механиката на играта,

покривајќи аспекти како балансирање на играта, системи за наградување, механика на прогресија и интеракција помеѓу механиката и однесувањето на играчите. Со користење на модели на дизајн во развојот на играта, како што е објектно-ориентиран дизајн, дизајн базиран на компоненти и дизајн управуван од настани, дизајнерите на игри можат да го оптимизираат искуството за играње. Со спојување на концептите за психологија на играчите во дизајнот, механиката и раскажувањето приказни, игрите можат да постигнат повисоки нивоа на ангажираност, задоволство и успех на играчите.

Еве неколку техники кои може да се применат:

- Динамично приспособување на тежината: имплементирање на системи кои динамички ја прилагодуваат тежината на играта врз основа на нивото на вештина и перформансите на играчот. Ова осигурува дека и почетниците и искусните играчи постојано се предизвикуваат, што доведува до задоволувачко искуство за сите.
- Приспособување според играчите: Обезбедува на играчите можност да го приспособат своето искуство во играта. Ова може да вклучува приспособливи аватари, ставки во играта или дури и прилагодување на темпото на играта. Таквото прилагодување им овозможува на играчите да ја прилагодат играта според нивните преференции.
- Персонализирање на наративот: Дизајнирање на разгранети наративи или повеќекратни патеки на приказни кои им овозможуваат на играчите да направат значаен избор, влијаејќи на приказната и исходот на играта. Оваа техника се грижи за играчите мотивирани од истражување и влијателно донесување одлуки.
- Карактеристики на социјална интеракција: Интегрирање социјални карактеристики, како што се режими за повеќе играчи, табли на водачи и заедници во играта, за да се погрижат на играчите кои бараат социјални интеракции и конкуренција.
- Достигнувања и награди: Имплементирање добро избалансиран систем на награди кој нуди достигнувања, значки или награди во играта за исполнување на предизвиците. Ова им се допаѓа на играчите мотивирани од чувството за достигнување и напредок.

- Истражување и откривање: дизајнирање обемни и детални светови на игри кои ги поттикнуваат играчите да истражуваат и откриваат скриени тајни. Играчите кои се мотивирани од љубопитност и истражување ќе бидат вовлечени во таквата игра.
- Емоционална резонанца: Развивање на ликови со релаксирачки емоции и мотивации и креирање приказни што предизвикуваат емоционални одговори од играчите. Емоционалниот ангажман може значително да го подобри доживувањето.
- Стимулација на сензори: користење аудиовизуелни елементи, вклучувајќи висококвалитетна графика, звучни ефекти и музика, за да создадете средина богата со сензори што играчите длабоко ќе се соживеат во светот на играта.
- Дизајн на состојба на проток: Дизајнирање на нивоата и предизвиците кои обезбедуваат оптимална рамнотежа помеѓу нивото на вештина на играчот и тежината на играта. Постигнувањето на оваа рамнотежа може да ги доведе играчите во состојба на „проток“, каде што се целосно апсорбирани и вклучени во играта.
- Управување со времето и темпото: Внимателно контролирање на темпото на играта, нудејќи моменти на интензитет проследени со периоди на релаксација. Оваа техника го одржува интересот на играчите и го спречува изгорувањето.
- Мајсторство и развој на вештини: Обезбедување на можности за играчите да ги развијат своите вештини и да ја совладаат сложената механика со текот на времето. На играчите мотивирани од самоподобрување и учење ќе им се допаднат ваквите предизвици.
- Повратни информации и интеракција: Понудување на непосредни и значајни повратни информации за дејствата на играчите. Одговорните контроли, визуелните знаци и хаптичките повратни информации го подобруваат чувството за поврзаност помеѓу играчот и светот на играта.
- Мистерија и интрига: Вклучување на мистерии, загатки или енигматични елементи кои ги интригираат играчите и ги мотивираат да истражуваат и да откријат скриени аспекти на играта.

Со комбинирање на овие техники и нивно приспособување на специфични типови и мотивации на играчи, игрите засновани на Unity можат да понудат разновидни и извонредни искуства кои резонираат со широк опсег на играчи.

3.4 Дискусија

Делот за дискусија навлегува во сеопфатна анализа на интеграцијата на основните принципи за развој на игри со платформата Unity, нудејќи увид во нејзините силни страни, ограничувања и потенцијални импликации за оваа област. Оваа синтеза ги премостува теоретските концепти со практична примена, подобрувајќи го нашето разбирање за тоа како овие принципи се синергизираат во околината на Unity.

Во неговото јадро, спојувањето на теоријата за дизајн на игри со Unity го олеснува беспрекорното преведување на теоретските рамки во опипливи искуства за играње. Вградувањето на механиката на играта, дизајнот на нивоа и принципите за ангажирање на играчите ги искористуваат можностите на Unity да конструира интерактивни светови што ги воодушевуваат играчите. Интерфејсот на Unity, вклучително и , Game View и Инспектор, се покажува како инструмент за визуелизација и дотерување на овие елементи, овозможувајќи им на програмерите повторливо да ги усовршуваат нивните дизајни.

Врската помеѓу психологијата на играчите и развојот на игри водени од Unity открива интригантна динамика. Интеракцијата помеѓу мотивацијата, типовите и задоволството на играчите, заедно со длабокото влијание на емоциите врз играта, наоѓа резонанца во доменот на Unity. Како што Unity продолжува да се развива, неговиот потенцијал дополнително да се усогласи со различни типови играчи и емоционални одговори се појавува како волшебна авенија за истражување.

Понатаму, техниките на раскажување приказни, како што се наративната структура, развојот на карактерот и градењето на светот, беспрекорно се испреплетуваат со можностите на Unity. Функциите за интерактивно раскажување приказни понудени од Unity им овозможуваат на програмерите да создаваат динамични наративи, каде изборот на играчите сложено ги обликува приказните што се развиваат. Оваа интеграција, сепак, поттикнува размислувања за степенот до кој Unity може да прими сложени наративни структури и навистина извонредни раскажувачки искуства.

Во тандем со теоретските димензии, прецизната анализа на механиката на играта во рамките на Unity ги нагласува нејзините силни страни и можните

ограничувања. Двојноста на балансирањето на играта, системите за наградување и механиката на прогресија наоѓа приемчиво платно во Unity, подобрувајќи ја динамиката на играта. Сепак, предизвикот лежи во постигнување хармонична интеракција помеѓу механиката и однесувањето на играчите, што може да бара дополнително префинетост и приспособливост во комплетот алатки на Unity.

Интеграцијата на дизајнерски шеми, како што е објектно-ориентиран, дизајн базиран на компоненти и дизајн управуван од настани, во екосистемот на Unity ја зајакнува разновидноста и растегливоста на погонот. Како и да е, бидејќи Unity приспособува различни шеми, мора да се разгледа како таквите шеми се усогласуваат со специфичните цели за дизајн на играта и дали одредени обрасци би можеле да бидат попогодни за архитектурата на Unity од другите.

Во суштина, интеграцијата на принципите за развој на игри со Unity најавува перспективна граница, каде теоретските конструкции и практичната имплементација хармонично се испреплетуваат. Дискурсот околу интерфејсот на Unity, управувањето со средства, составот на сцената, анимацијата и скриптата го нагласува неговиот потенцијал како креативен катализатор, а истовремено навестува области за потенцијален раст и оптимизација. Како што способностите на Unity продолжуваат да се развиваат, ова истражување нè позиционира ефикасно да ја искористиме неговата моќ, збогатувајќи го пејзажот на развојот на играта и поттикнувајќи поприврлечни искуства за играчите.

IV. БЕТА- ИГРА

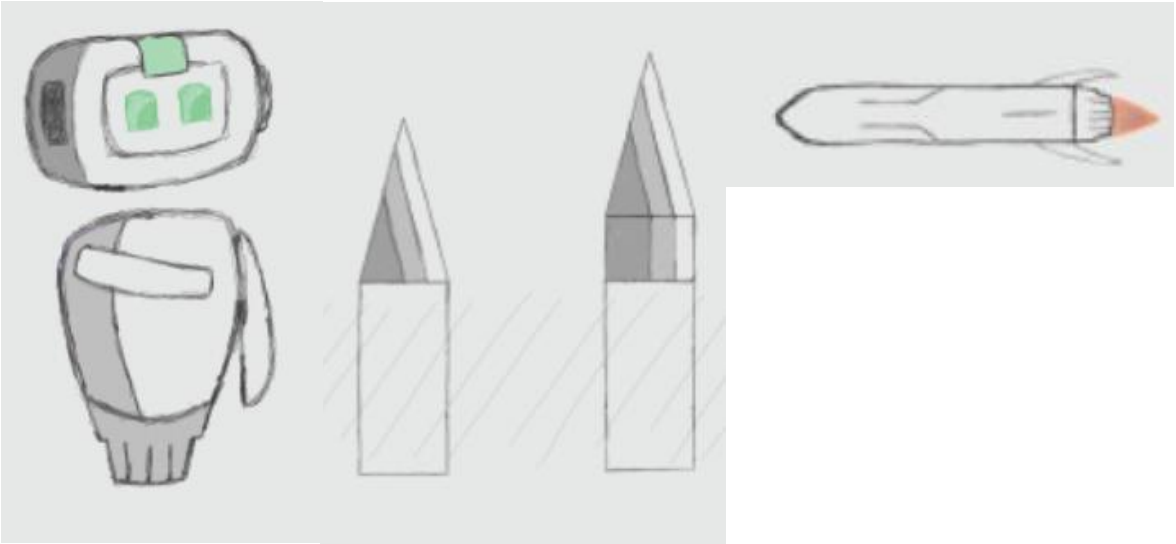
4 КОНЦЕПТУАЛИЗАЦИЈА И ДИЗАЈН

Овој поддел се фокусира на фазата на концептуализација и дизајн на играта. Се дискутира за иницијалниот процес на генерирање идеи, идентификацијата на целната публика и целите на играта, како и развојот на основниот концепт и механиката. Може да вклучува детали за сесии за бура на идеи, концептни скици и рани повторувања на дизајнот.

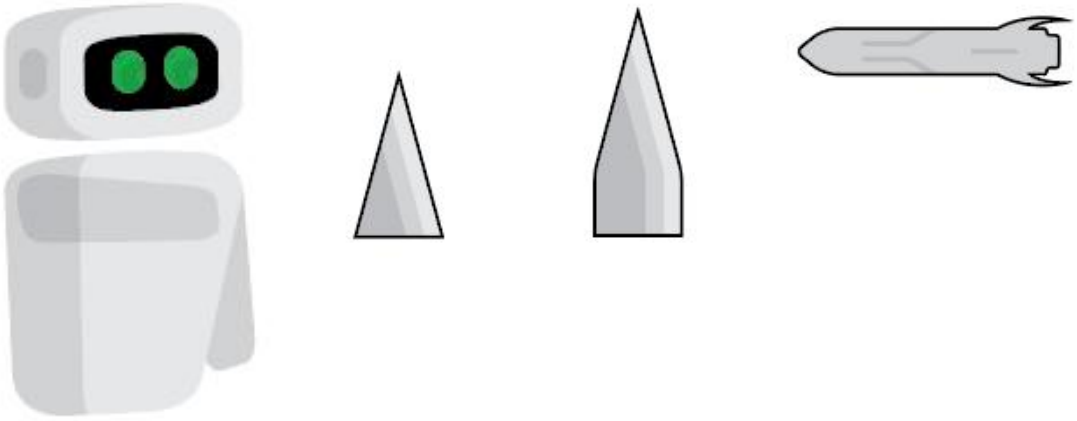
- Бура на идеи

Мојата бура на идеи вклучуваше игра од типот бескраен тркач, каде карактерот е робот кој се обидува да ги избегне препреките, главниот фокус беше создавање едноставно, но волшебно искуство, совршено за лежерни моменти. Црпејќи инспирација од класиците како Flappy Bird и играта со диносаурус на Google Chrome, целта беше да се дизајнира игра која без напор ќе ги пополни тие слободни моменти низ времето, нудејќи им на играчите спокојно и пријатно опуштање.

- Концептни скици

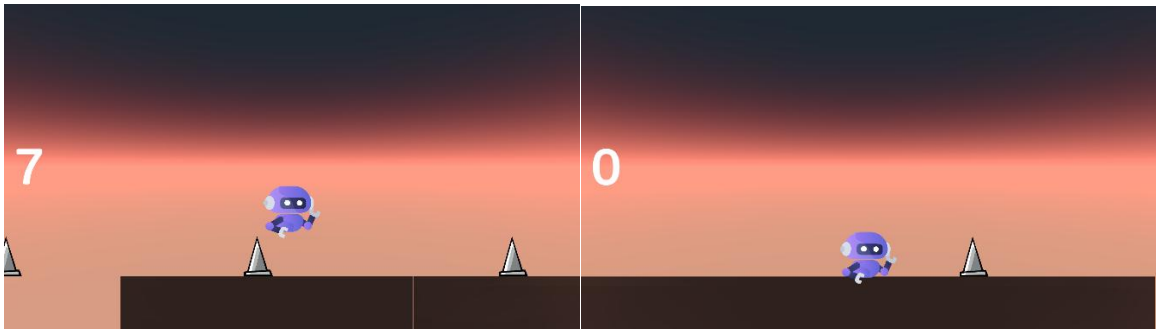


4- 1 Почетна концепт скица за карактер и пречки



4- 2 Трансформирање на скицата во полиран дизајн на карактерот и пречките со помош на Illustrator

- Ран дизајн



4- 3 Ран дизајн на играта

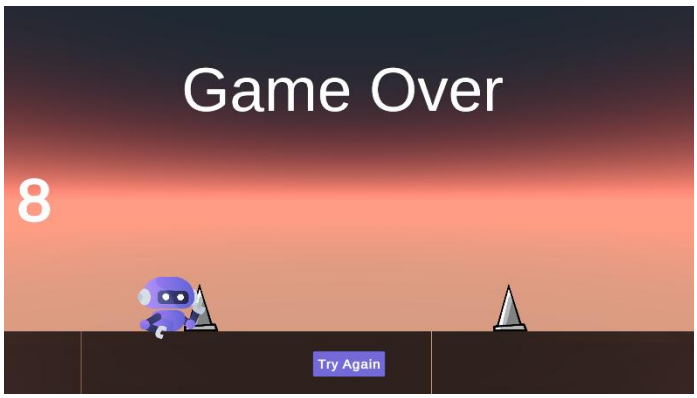
Во раните фази на развојот, пред да биде дизајниран финалниот модел на карактерот и пречките беа користени привремени средства исклучиво за визуелно водство. Тоа ми овозможи да ја поставам основата за мојата визија, нудејќи груб увид во идниот потенцијал на играта.

4.1 Концепт на играта и механика

Оваа потточка навлегува подлабоко во конкретниот концепт на играта и механиката на играта. Ги опишува основните карактеристики на играта, контролите и интеракциите кои го обликуваат искуството на играчот. Може да вклучува детали за поставките на играта, ликовите, целите, нивоата, системите за прогресија и сите уникатни елементи на игра.

- Основни карактеристики на игра

Основните карактеристики на играта се вртат околу концептот на бескраен тркач со протагонист робот кој ги избегнува пречките. Два различни режими се достапни на почетокот на играта. Во режимот 1, играчите го водат роботот низ рандом поставените видни празнини, при што роботот автоматски се спушта и се искачува по притискање на допир (за мобилен) или притискање на празно место (Space) (за компјутер). Успешното поминување низ празнина заработува еден поен, но судирот со ѕидовите или земјата ја завршува играта и го прикажува најдобриот резултат до тој момент. Режимот 2 претставува слична премиса, но роботот ги прескокнува пречките и се доделуваат поени за нивно успешно расчистување. Додека бета верзијата е за еден играч, идните развојни планови вклучуваат имплементација на табла на водачи за подобрување на



конкурентниот аспект. Уникатната комбинација на навигациска механика и системи за бодување на играта ја прави привлечно и динамично искуство за играчите.

- Контроли

Играта има директни контроли за да се обезбеди пристапност и леснотија на игра. За мобилните уреди, играчите можат да го допрат екранот за да го натераат роботот да се искачи, додека за компјутер, едно притискање на празно место го постигнува истиот ефект. Овие интуитивни контроли ја подобруваат пристапноста на играта, правејќи ја пријатна за играчите од сите нивоа на вештини.

- Детали
    - Поставки на играта

Играта е сместена во футуристички свет со посебна естетика, им нуди на играчите извонредно искуство на егзотични планети, додавајќи визуелно разновидна димензија на играта.

    - Ликови

Главниот лик на играта е елегантен, бел и сив робот со експресивни зелени „очи“ прикажани на неговиот црн екран. Овој робот служи како аватар на играчот во текот на играта.

    - Цели

Во оваа игра, примарна цел е да се забавувате и да се стремите да ги победите личните високи резултати. Бета- верзијата се фокусира на предизвикување на играчите да ги надминат сопствените рекорди, додека идниот развој ќе воведе табли на водачи за компететивно натпреварување.

    - Нивоа

Играта следи бесконечен формат на тркач, што значи дека нема различни нивоа. Сепак, играчите можат да очекуваат промени на сценографијата во позадина за да се одржи визуелното искуството привлечно и динамично.

    - Системи за прогресија

Иако бета- верзијата нема систем за прогресија, идните развојни планови вклучуваат воведување продавница во играта. Овде, играчите можат да ја трошат својата валута во играта, заработена преку играње, за да отклучат нови изгледи на роботот и други додатоци, овозможувајќи персонализирано прилагодување.

- Уникатни елементи на игра

Додека основната игра се фокусира на класичната механика бескрајни тркачи, уникатните опции за прилагодување на карактерите на играта во идната верзија ќе им дадат на играчите можност да го стилизираат и персонализираат својот робот, додавајќи карактеристичен и привлечен елемент на искуството.

4.2 Размислувања за дизајн и кориснички интерфејс

Оваа потточка се фокусира на размислувањата за дизајнот и корисничкиот интерфејс (UI) за играта. Се дискутира за визуелниот стил, уметничката насока, шемите на бои и севкупните естетски избори кои ја подобруваат атмосферата на играта и корисничкото искуство. Може да се однесува и на принципите за дизајн на интерфејс, како што се распоред, навигација, системи за повратни информации и размислувања за пристапност. Дополнително, може да се дискутира за употребата на алатките и техниките за интерфејс на Unity за имплементирање на саканиот дизајн и елементи на интерфејсот.

- Визуелен стил

Играта прифаќа футуристички визуелен стил, комбинирајќи елегантни, метални елементи со високотехнолошка естетика лоцирани на некоја егзопланета. Овој стил предизвикува чувство на напредна технологија и роботика, усогласувајќи се со целокупната тема на играта.

- Уметничка насока

Уметничката насока за играта нагласува чист и минималистички пристап. Фокусот е на создавање визуелно привлечни, лесно препознатливи елементи кои го подобруваат доживувањето на играчот во футуристичкиот свет на играта.

- Шеми на бои

Избраните шеми на бои претежно се карактеризираат со студени, модерни тонови, како што се нијанси на сива, бела и металик- нијанси. Овие бои се исцртани со живописни зелени, особено во „очите“ на роботот, додавајќи блесок на бои и истакнувајќи важни елементи во играта.

- Севкупни естетски избори

Естетските избори на играта имаат приоритет на хармоничен спој на високотехнолошки футуризам и дизајн лесен за корисниците. Овие избори придонесуваат за привлечно и визуелно волшебно искуство за играчите.

- Принципи за дизајн на UI
    - Распоред

Распоредот на корисничкиот интерфејс (UI) е дизајниран да биде интуитивен и ненаметлив, осигурувајќи дека основните информации се лесно достапни без да се натрупува екранот.

    - Навигација

Навигацијата во интерфејсот е погодна за корисниците, дозволувајќи им на играчите да се префрлаат помеѓу режимите на игра, да пристапуваат до поставките итн.

    - Системи за повратни информации

Интерфејсот вклучува јасни системи за повратни информации за да ги информира играчите за нивниот напредок, резултат и која било валута во играта што ја заработиле(во идни верзии).

4.3 Имплементација на играта со помош на Unity

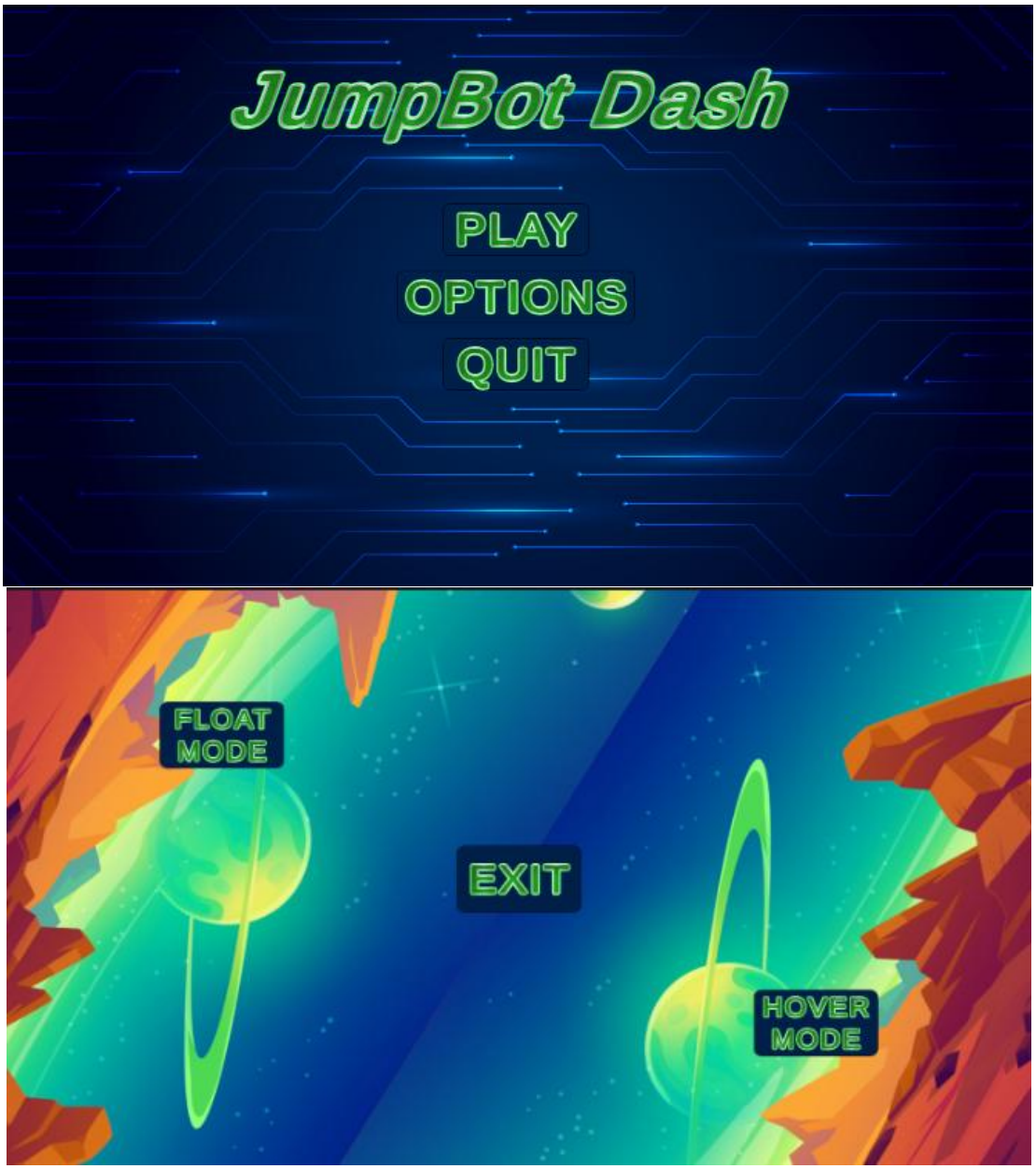
Оваа потсекција истражува за вистинската имплементација на играта користејќи ја платформата за развој на игри Unity. Го дискутира процесот чекор по чекор на трансформирање на концептот и дизајнот на играта во искуство за играње во Unity. Може да опфати аспекти како што се внес на средства, создавање и

конфигурирање на објекти на играта, имплементација на механика за играње, интегрирање аудио и визуелни елементи и оптимизирање на перформансите.

- Процес чекор по чекор

Создавање на сцена: Првично, започнав со создавање на различни сцени, вклучувајќи го главното мени, менито со опции, мени за избор на режим, режимот

4- 4 Приказ на мени-ата во играта



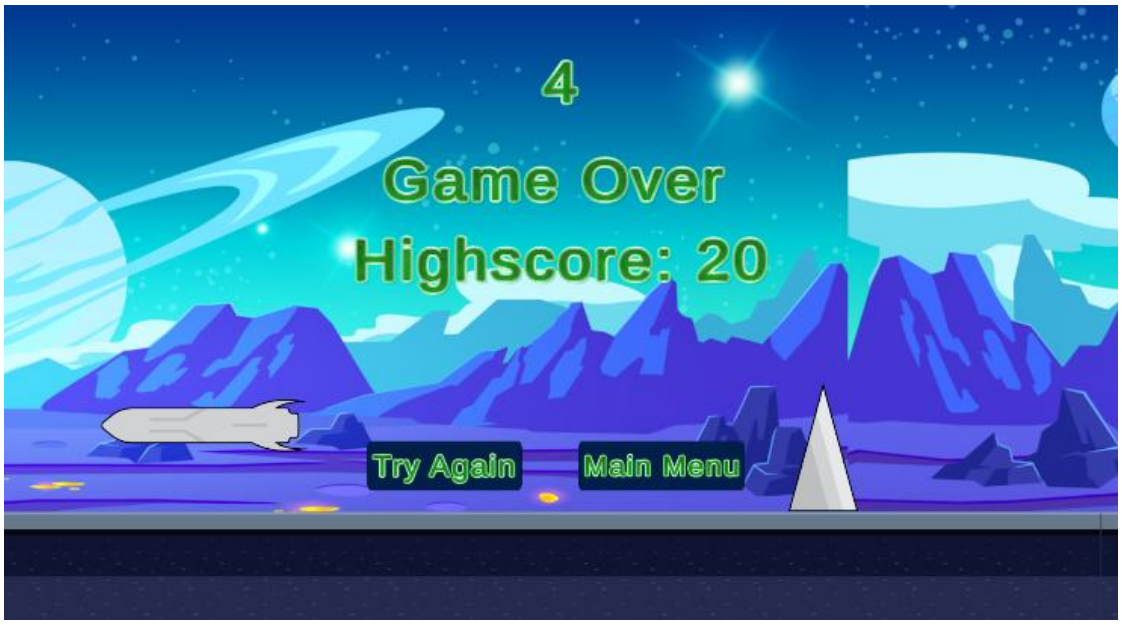
на игра 1 и режимот на игра 2.

Создавање на лик (робот): Во режимот на игра 1, го дизајнирав и креирав главниот лик, робот. Ги прикачив основните компоненти како што се колајдер, Rigidbody2D и скрипти за да се контролира неговото однесување.

Создавање на пречки: Продолжив со создавање на пречките во играта. Тие беа претворени во префабрикувани производи за лесна репликација. Во меѓувреме ги развивав скриптите на објектите во играта како генерирање на пречки за време на играта, логика итн.

Ефекти на позадина: За да се подобри искуството во играта, имплементирав ефект на подвижна позадина, додавајќи динамика на сцените.

Интерфејс: Создадов кориснички интерфејс (UI) што обезбедува опции за играчите да се обидат повторно или да се вратат на менито за избор на режим. Интерфејсот вклучува копчиња и скрипти за да ја олесни навигацијата низ сцените. Дополнително, имплементирав систем кој зачувува највисокиот резултат.



4- 5 Приказ на прозорецот кога играта ќе заврши и опциите за враќање во менито ли повторно играње

Режим на игра 2: Го реплицирав процесот за режимот на игра 1, вклучувајќи креирање карактери, поставување пречки, ефекти во заднина и интерфејс.

Аудио интеграција: За дополнително ангажирање на играчите, додадов аудио елементи. Ова вклучуваше музика во позаднина во менијата и звучни ефекти за интеракции како што е роботот кој минува низ пречки и добива поени или се судира со пречките.

- Внес на средства

За овај проект внесов неколку средства кои ми помогнаа, го олесниј а и забрзаа процесот на креирање на играта, како што се аудио- фајловите.

- Механика

Мрестење на пречки: Развив систем за мрестење на пречките контролиран со скрипта. Оваа механика обезбедува континуиран предизвик за играчите бидејќи пречките се создаваат во случајни интервали и позиции во светот на играта.

Систем за бодување: Системот доделува поени на играчот секој пат кога роботот успешно ја минува пречката.

- Оптимизирање на перформансите

За оптимизација на играта објектите кои се мрестат кога ќе излезат од приказот на сцената, се бришат за ослободување на меморијата.

4.4 Тестирање

После секое креирање на објект и секоја скрипта спроведував тест за тоа дали правилно се однесува играта како што беше замислено. Сите опции, копчиња и карактеристики беа тестирани повеќе пати за нивната исправност.

4.5 Идни планови

Во иднина многу аспекти на играта можат да бидат подобрени, а истовремено може да се спроведат и нови карактеристики како нови режими на игра, во менито за поставки да се спроведат опции за резолуција, јазик, опција за промена на позадините, музиката, нови препреки, табла со водачи на највисок резултат, анимации, итн. Концептот за играта како и многу познати игри според мое мислење е добар за релаксација и занимација во слободното време.

ЗАКЛУЧОК

Овај труд навлезе во различни аспекти на дизајнот на играта, развојот и ангажирањето на играчите, нагласувајќи го нивното значење во создавањето извонредни и волшебни искуства во играта. Механиката на играта служи како основни градежни блокови на играта, обликувајќи ги интеракциите и искуствата на играчите. Правилно дизајнираната механика, интуитивна и усогласена со целите на играта е од суштинско значење за привлечно искуство во играта.

Дизајнот на нивоа беше дискутиран како повеќедимензионален процес, нагласувајќи ја важноста на елементите како структура, рамнотежата на тежината, можности за истражување, тематска естетика и итеративна префинетост во создавањето воодушевувачки нивоа на играта. Овие елементи колективно придонесуваат за ангажирање на играчите, задржувајќи го нивниот интерес во текот на играта.

Ангажирањето на играчите, клучен аспект на дизајнот на играта, се врти околу привлекување на вниманието на играчите, предизвикување емоции и обезбедување чувство на агенција и исполнетост. Програмерите може да го постигнат ова преку внимателен дизајн на игри, значајно раскажување приказни, социјална интеракција и редовни ажурирања, кои помагаат да се негува посветена и ентузијастичка база на играчи за долгорочен успех.

Беше истражено разбирањето на психологијата на играчите и нејзиното влијание врз нивното однесување и искуството во игрите, со теми како мотивација, типови на играчи, задоволство, доживување и емоционално влијание. Ова разбирање може да ги поттикне развивачите на игри да создадат попривлечни искуства преку искористување на психолошките теории и модели.

Беше разгледана и улогата на наративот во игрите, фокусирајќи се на наративните структури, развојот на ликови, градењето на светот и интерактивното раскажување приказни. Ефективното раскажување приказни го подобрува ангажманот и доживувањето на играчите, додавајќи длабочина на искуството со играњето.

Беше анализирано и влијанието на механиката на играта врз играта и искуството на играчите, опфаќајќи теми како балансирање на играта, системи за наградување, механика на прогресија и нивната интеракција со однесувањето на играчите. Овие механики значително влијаат на одлуките на играчите, развојот на вештините и целокупното уживање.

Понатаму, трудот ги истражуваше заедничките модели на дизајн и најдобрите практики во развојот на игри, нагласувајќи ја нивната важност за подобрување на организацијата на кодот, модуларноста и одржливоста во проектите за развој на игри.

Методологијата што се користи за да се запознаете со платформата за развој на игри Unity, нејзините карактеристики, управување со средства, состав на сцена и техники за скриптирање и кодирање беа детално дискутирани, обезбедувајќи цврста основа за оние кои влегуваат во полето на развој на игри.

При практичната примена на знаењето стекнато низ овај труд, принципите и концептите истражени овде одиграа клучна улога во развојот на бета играта. Дополнително, длабинското познавање на карактеристиките на Unity, управувањето со средства, составот на сцената и методологиите за скриптирање, стекнато во овај труд, послужи како основа за техничкото извршување на проектот. Оваа успешна интеграција на теоретските концепти во практичниот развој на игри ја нагласува непроценливата улога на овај труд во обликувањето на успехот на бета играта.

КОРИСТЕНА ЛИТЕРАТУРА

[1] Aarseth, E. (2003). Computer game studies yearbook 2002. MIT Press.

[2] Aarseth, E. J. (2004). Computer Game Studies. MIT Press.

[3] Adams, E. (2014). Fundamentals of game design (2nd ed.). San Francisco, CA: Addison-Wesley.

[4] Aldrich, C. (2005). Realism in Serious Games: Embracing Play for a Purpose.

[5] Amory, D., & Boyd, D. (2008). The design of playful experiences. MIT Press.

[6] Ampatzoglou, Apostolos & Chatzigeorgiou, Alexander. (2007). Evaluation of object-oriented design patterns in game development. Information and Software Technology. 49. 445-454. 10.1016/j.infsof.2006.07.003.

[7] Andresen, H. (2003). Component-based design in game development. GameDev.net, 2(4).

[8] Apperley, T. (2007). Cinematic techniques for game designers and artists: Using the grammar of film to create compelling visual stories. Focal Press.

[9] Apperley, T. (2007). Genre and narrative in video games. In J. Whitehead & M. Saarikoski (Eds.), The Routledge companion to video game studies (pp. 207-223). Routledge.

[10] Apperley, T. (2019). Worldbuilding for games: A practical guide to creating immersive and engaging game worlds. Routledge.

[11] Apter, M. (2008). The Book of Balance: Strategies for Game Design. Charles River Media.

[12] Bandura, A. (1997). Self-efficacy: The exercise of control. W.H. Freeman.

[13] Bartle, R. A. (1996, February). Hearts, Clubs, Diamonds, Spades: Players Who Suit MUDs. In MUDs in Action (pp. 73-89). John Wiley & Sons.

[14] Bateson, M. (2001). Designing Games: A Guide to Creating Engaging Experiences. Pearson Education.

[15] Booch, G. (1994). Object-oriented analysis and design with applications (3rd ed.). Addison-Wesley.

[16] Buttfield-Addison P., Jonathon Manning, Tim Nugent, (16 April, 2019). Unity Game Development Cookbook: Essentials for Every Game (1st ed.)

[17] Bycer J., How Event-Driven Game Design Keeps the Player Guessing. https://game-wisdom.com/critical/event-driven-game-design

[18] ChunTing Wu, Design Patterns of Event-Driven Architecture. https://dev.to/lazypro/design-patterns-of-event-driven-architecture-3a79

[19] Crawford C., (01 December, 2002). The Art of Interactive Design: A Euphonious and Illuminating Guide to Building Successful Software (1st ed.)

[20] Crawford, C. (2004). Interactive storytelling. Digital game studies reader, 159-175.

[21] Crawford, C. (2003). The Art of Computer Game Design. O'Reilly Media.

[22] Crawford, C. (2009). Super Mario Bros. 3: An Analysis of Game Mechanics. https://www.significant-bits.com/super-mario-bros-3-level-design-lessons/

[23] Csikszentmihalyi, M., & LeFevre, M. (1991). Flow: The psychology of optimal experience. HarperCollins Publishers Inc.

[24] De Castellar, E. M. (2009). The rise of network play: A cultural and technological history of video games. MIT Press.

[25] Deci, E. L., & Ryan, R. M. (2000). The intrinsic motivation to engage in self-determined behavior. In Handbook of self-regulation: Motivation, volition, and action control (pp. 189-232). Guilford Press.

[26] Deterding, S., Dixon, D., & Khaled, R. (2011). Gamification: Using game-design elements in non-game contexts. Digital Press.

[27] Deterding, S., Dixon, D., & Lehner, P. (2011). Gamification: Using game-based mechanics to engage and motivate. O'Reilly Media.

[28] Eckel, B., & Stroustrup, B. (2019). Programming: Abstraction, algorithms, and data structures using C++. Addison-Wesley.

[29] Egenfeldt-Nielsen S., Jonas Heide Smith, Susana Pajares Tosca, (24 September, 2019). Understanding Video Games (4th ed.)

[30] Epps, B. W. (2009). Collaborative game-based learning: A model for integrating technology, pedagogy, and assessment. ISTE (Instructional Technology, Inc.).

[31] Erickson, G., & Katajainen, M. (2005). Object-oriented design with Java using UML 2.2 (2nd ed.). Addison-Wesley.

[32] Eskelinen, M. (2004). Beyond the three act structure: A flexible narrative framework for interactive stories. In H. I. Korhonen, T. Keinonen, & P. Myllyniemi (Eds.), DiGRA 2003: Proceedings of the 2003 Digital Games Research Association Conference (pp. 1-11). Tampere Game Research Institute.

[33] Estevez Andrés Alberto , A component-based architecture for Unity. https://medium.com/etermax-technology/a-component-based-architecture-for-unity-ff211ca478fe

[34] Fabricatore, Carlo. (2007). Gameplay and game mechanics design: a key to quality in videogames. 10.13140/RG.2.1.1125.4167.

[35] Ferrone Harrison, (29 October, 2021). Learning C# by Developing Games with Unity 2021, Kickstart your C# programming and Unity journey by building 3D games from scratch, (6th ed.)

[36] Foss, M. (2005). Character development for writers. Heinemann.

[37] Gaiser, T. (2004). Narrative in games: The essentials for story designers. Peachpit Press.

[38] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). Design patterns: Elements of reusable object-oriented software. Addison-Wesley.

[39] Garris, R., Ahlers, R., & Driskell, J. D. (2002). Games and learning: Research and practice. Pearson College Div.

[40] Gentile, D. A. (2009). Pathological video game use among youth: A comparative and international perspective. Psychological Bulletin, 135(5), 722-731.

[41] Gestwicki, P. and Sun, F.-S. 2008. Teaching design patterns through computer game development. ACM J. Educ. Resour. Comput. 8, 1, Article 2 (March 2008), 21 pages. DOI = 10.1145/1348713.1348715. http://doi.acm.org/10.1145/1348713.1348715.

[42] Gregory, Michael (2005). Game engine architecture: A practical guide for real-time game development. Charles River Media, Inc.

[43] Habgood, J. P. (2008). Level design for mobile games. Jones & Bartlett Learning.

[44] Hamari, J. (2012). The psychology of gamification. American Psychological Association.

[45] Hamari, J. (2014). Why do we play video games? A motivational analysis.

[46] Hamari, J., & Koivisto, J. (2015). From novelty to addiction: Exploring the motivational effects of gamification. Computers in Human Behavior, 49, 185-190.

[47] Hamari, J., Järvelä, S., & Takala, M. (2014). Motivational components of gamification: A systematic empirical review. Motivation and Science, 2(1), 1-33.

[48] Hamari, J., Shuldbraut, J., & Sukthankar, G. (2014). An exploration of user experience and quality in mobile free-to-play games. Computers in Human Behavior, 35, 240-248.

[49] Hennig, C., Vorderer, P., & Roth, K. (2017). Emotions and video games: A review of the literature. Entertainment Computing, 20, 1-23.

[50] Hocking J., (25 April, 2018). Unity in Action, Multiplatform Game Development in C# (2nd ed.)

[51] Hoffer, K., & Ricco, M. (2011). The narrative toolbox for writers and filmmakers: A guide to story structure, characters, conflict, and dialogue. Focal Press.

[52] Jackson S.(26 August, 2014). Mastering Unity 2D Game Development

[53] Jenkins, H. (2004). The transmedia storytelling reader. Routledge.

[54] Jennett, C., Heron, S., & Calvert, T. (2008). Measuring and defining presence in virtual worlds. In Presence: Multisensory issues (pp. 109-140). Springer, London.

[55] Jenova, P. (2015). Immersion in video games: Towards a conceptual model. Frames: Cinema Journal, 10(1), 30-39.

[56] Kapp, K. M., & Kim, Y. (2014). Learning in games: A guide for educators and designers. New York: Routledge.

[57] Klimmt, Christoph & Blake, Christopher & Hefner, Dorothee & Vorderer, Peter & Roth, Christian. (2009). Player Performance, Satisfaction, and Video Game Enjoyment. 5709. 1-12. 10.1007/978-3-642-04052-8_1.

[58] Koepke, M. (2014). Journey: A video game analysis.

[59] Koster R., (31 December, 2013). Theory of Fun for Game Design (2nd ed.)

[60] Larman, C. (2004). Applying UML and patterns: An object-oriented introduction to programming. Addison-Wesley.

[61] Leblanc, M. (2004). Narrative design: Essential concepts for creating story worlds. Peachpit Press.

[62] Lebowitz J., Chris Klug, (01 March, 2011). Interactive Storytelling for Video Games: A Player-Centered Approach to Creating Memorable Characters and Stories (1st ed.)

[63] Leder, M., & Deci, E. L. (2008. Conscientiousness and motivation: The role of self-regulatory processes. Personality and Social Psychology Bulletin, 34(10), 1476-1491.

[64] Lee, J., & Kang, H. W. (2009). Object-oriented programming and game development: A practical guide*. CRC Press.

[65] Lennartsson, P. (2013). The interplay between player behavior and game mechanics. In The Handbook of Game Studies (pp. 349-364). MIT Press.

[66] Levy L., Jeannie Novak, (22 June, 2009). Game Development Essentials: Game QA & Testing (1st ed.)

[67] Locke, E. A., & Latham, G. P. (1990). A theory of goal setting and task performance. Englewood Cliffs, NJ: Prentice Hall.

[68] Loftus, J. W. (2003). Why we remember what we do: Memory, consciousness, and the brain. Houghton Mifflin Harcourt.

[69] Lombard, M., & Ditton, T. (2007). At the heart of play: An overview of intrinsic and extrinsic motivations in video games. In J. P. Hanus & M. O. Mäystö (Eds.), Applied psychology of video games (pp. 129-150). Lawrence Erlbaum Associates.

[70] Lumsden, J., & Millar, C. (2008). The game design reader. MIT Press.

[71] Madeira, Filipa & Arriaga, Patricia & Adrião, Joana & Lopes, Ricardo & Esteves, Francisco. (2013). Emotional Gaming.

[72] Malone, T. W. (1981). Heuristics for Designing Enjoyable Human-Computer Interfaces. https://dl.acm.org/doi/10.1145/800049.801756

[73] Malone, T. W. (1981). What makes things fun to do? Psychology of learning and motivation, 16(1), 127-153.

[74] Maxwell Chandler H., (20 March, 2013). The Game Production Handbook (3rd ed.)

[75] McCloud, S. (2003). Reinventing comics: How graphic novels (and video games) transcended the format. HarperCollins Publishers.

[76] McEachern, M., & Morgan, C. (2007). Component-based development for games: A practical guide. Addison-Wesley.

[77] McGloon, M. (2007). Designing virtual worlds. Pearson Education.

[78] McGreal, T. (2010). Challenge in Games: Using Failure to Enhance Play. Springer.

[79] McKeown, William M. (2011). Game development with C++: A practical guide to real-time game programming. CRC Press.

[80] Meyer, B. (1988). Object-oriented software construction. Prentice Hall.

[81] Moll, Philipp & Frick, Veit & Rauscher, Natascha & Lux, Mathias. (2020). How players play games: observing the influences of game mechanics. 7-12. 10.1145/3386293.3397113.

[82] Molyneux, P. (2014). The Last of Us: A review.

[83] Moore, M. E. (2011). Does the difficulty curve matter?. In M. E. Moore (Ed.), The game designer's handbook (pp. 176-184). Burlington, MA: Elsevier.

[84] Nakamura, J., & Csikszentmihalyi, M. (2002). Flow experience: A cognitive-affective model of subjective well-being. In C. Snyder & S. J. Lopez (Eds.), Handbook of positive psychology (pp. 305-322). Oxford University Press.

[85] Novak, J. (2005). Balancing game elements: How game mechanics and design affect player choice. New York, NY: Focal Press.

[86] Nystrom R., (02 November, 2014). Game Programming Patterns

[87] Oliver, M. (2007). Creating immersive experiences for learning. In Handbook of research on learning and instruction (pp. 455-472). Routledge.

[88] Официјална страна за упатсво на уботреба на карактеристиките на Unity. https://docs.unity3d.com/Manual/index.html

[89] Petrus, M., & Bogost, I. (2009). Putting the player in play: A framework for understanding digital game player participation. In Proceedings of the 42nd ACM conference on Computers in play (pp. 397-400).

[90] Prensky, M. (2001). Digital GameDesign and Learning. MIT Press.

[91] Prensky, M. (2001). Digital natives, digital immigrants part 1. On the Horizon, 9(5), 1-6.

[92] Przybylski, A., Rigby, C., & Ryan, R. M. (2012). Motivational typology of video game engagement. Psychological bulletin, 138(2), 219-246.

[93] Przybylski, A., Rigby, C., & Ryan, R. M. (2013). A motivational science perspective on the use and perceived effects of video games. Motivation and Emotion, 37(1), 1-16.

[94] Przybylski, R., & Ryan, R. M. (2010). Psychological benefits of playing video games. American Psychologist, 65(1), 67-71.

[95] Przybylski, R., Rigby, C., & Ryan, R. M. (2006). Motivational typologies of adult video game users. Journal

[96] Przybylski, R., Rigby, C., & Ryan, R. M. (2014). Motivational typologies of video game engagement. Motivation and Science, 3(2), 190-206.

[97] Rogers S., (28 April, 2014). Level Up! The Guide to Great Video Game Design (2nd ed.)

[98] Rouse, M. (2009, October 29). Response Time. https://www.techopedia.com/topic/47/gaming

[99] Rouse, M. (2020, April 20). What Is Frame Rate? https://www.techopedia.com/definition/3036/frame-rate

[100] Ryan, M. E. (2001). Beyond built-in fun: Narrative integration in digital games. MIT Press.

[101] Ryan, M. L. (2004). Avatars of narrative: Interactive storytelling in video games. Iowa State Press.

[102] Ryan, R. M. (2010). Toward a psychological theory of motivation and emotion: The motivational pyramid model. In R. M. Ryan (Ed.), The handbook of self-determination research (pp. 35-67). Rochester, NY: Rochester University Press.

[103] Ryan, R. M., & Deci, E. L. (2000). Intrinsic and extrinsic motivations: Classic definitions and new directions. Contemporary Educational Psychology, 25(1), 54-67.

[104] Ryan, R. M., Korhonen, J., & Lin, Y. (2006). Energizing and depleting forms of self-regulation in motivation. Psychological Bulletin, 132(2), 230-249.

[105] Sailer, M., Pemsel, H., & Bogge, T. (2017). Player experience in game progression: Balancing pacing, difficulty, and choice. In A. Raessens & C. D. Martens (Eds.), Handbook of game design (pp. 1-39). Cambridge University Press.

[106] Salen, K., & Zimmerman, E. (2004). Rules of Play. MIT Press.

[107] Sawyer, B. (2007). Developing games: A beginner's guide. O'Reilly Media, Inc.

[108] Schell J., (04 August, 2008). The Art of Game Design: A Book of Lenses (1st ed.)

[109] Schoenau-Fog H., The Player Engagement Process–An Exploration of Continuation Desire in Digital Games. http://www.digra.org/wp-content/uploads/digital-library/11307.06025.pdf

[110] Schreck, J. (2016). The psychology of fear in video games: A review.

[111] Schreiber, I. (2009). Level design: Concepts and practice. San Francisco, CA: Morgan Kaufmann.

[112] Schroeder, R. (2011). Creating immersive virtual worlds: A practical guide to experiential design. Apress.

[113] Seabrook, M. (2011). Digital games and learning: The promise and the challenges. Palgrave Macmillan.

[114] Seabrook, R. (2014). Why we play games: Evolution, neuroscience, and the hidden joys of interaction. Penguin.

[115] Sharp, J., & Dittrich, Y. (2006). Easy to Learn, Difficult to Master: Multidisciplinary Methods in Game Design and Development. Lawrence Erlbaum Associates.

[116] Sheldon L., (18 November, 2022). Character Development and Storytelling for Games (3rd ed.)

[117] Sicart, M. (2007). Playing with Fears: The Appeal of Horror Games. MIT Press.

[118] Sicart, M. (2008). Playing with the world: How video games provoke agency. MIT Press.

[119] Sirlin, D. (2013). Playing to win: How game theory can make you a better player. San Francisco, CA: No Starch Press.

[120] Squire, K. (2004). Replaying history: Video games and the reenactment of the past. New York: Palgrave Macmillan.

[121] Stokes, A. (2010, October 12). Frame Rate and Response Time: The Basics. https://tasvideos.org/PlatformFramerates

[122] Swink S. (13 October, 2008). Game Feel: A Game Designer's Guide to Virtual Sensation (1st ed.)

[123] Sylvester T. (26 March, 2013). Designing Games: A Guide to Engineering Experiences (1st ed.)

[124] Szűcs, A., & Hoffer, J. (2015). Worldbuilding for video games. In The Routledge companion to video game studies (pp. 252-263). Routledge.

[125] Tamborini, R., Skalski, P., & Reeves, B. (2015). Interactivity and narrative: An examination of their role in immersive media experiences. Journal of Media Psychology, 27(2), 128-142.

[126] Tamimi A., A Guide for the Developer's Journey to Event-Driven Development. https://solace.com/blog/developer-guide-event-driven-development/

[127] The Level Design Book. https://book.leveldesignbook.com/

[128] Trevisan João Pedro , Literature Review The Psychology of Motivation in games.

[129] Wenderlich R., (01 January 2013). Introduction to Component Based Architecture in Games. https://www.kodeco.com/2806-introduction-to-component-based-architecture-in-games#toc-anchor-004

[130] Werker, J. F. (2008). Action video games and their potential impacts on youth. Oxford University Press.

[131] White, R. P., & Leasure, J. L. (2006) How to manage e-learners: A guide to motivation and engagement. San Francisco: Jossey-Bass.

[132] Whitty, R. W. (1997). Multimedia Learning: Combining Words and Images. John Wiley & Sons.

[133] Yee, N. (2006, November). The Motivational Pull of Videogames. In Proceedings of the 7th International Conference on Human-Computer Interaction (pp. 157-160). ACM.

[134] Zagal, J. (2019). The Art of Game Design: A Play-Centric Approach. CRC Press.

Прилог 1: Слики од играта



Прилог 2: Дел од Скрипти

```
Unity Script (3 asset references) | 0 references
public class VolumeManager : MonoBehaviour
{
    [SerializeField] Slider volumeSlider;

    Unity Message | 0 references
    private void Start()
    {
        if (!PlayerPrefs.HasKey("volume"))
        {
            PlayerPrefs.SetFloat("volume", 1);
        }
        else
        {
            Load();
        }
    }

    0 references
    public void ChangeVolume()
    {
        AudioListener.volume = volumeSlider.value;
        Save();
    }

    1 reference
    private void Load()
    {
        volumeSlider.value = PlayerPrefs.GetFloat("volume");
    }

    1 reference
    private void Save()
    {
        PlayerPrefs.SetFloat("volume", volumeSlider.value);
    }
}
```

```
[RequireComponent(typeof(AudioSource))]
Unity Script (2 asset references) | 4 references
public class PointAudioScript : MonoBehaviour
{
    private AudioSource audioSource;

    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {
        // Get the AudioSource component attached to this GameObject
        audioSource = GetComponent<AudioSource>();

        // Initially, disable the audio
        audioSource.enabled = false;
    }

    // Call this function to play the audio
    2 references
    public void PlayPointAudio()
    {
        // Enable the audio source and play the audio
        if (audioSource != null && !audioSource.isPlaying)
        {
            audioSource.enabled = true;
            audioSource.Play();
        }
    }
}
```

```
Unity Script (1 asset reference) | 1 reference
public class AudioManager : MonoBehaviour
{
    private static AudioManager instance;

    Unity Message | 0 references
    void Awake()
    {
        // Check if an instance of AudioManager already exists
        if (instance == null)
        {
            // If not, set this as the instance and make it persistent across scenes
            instance = this;
            DontDestroyOnLoad(this.gameObject);
        }
        else
        {
            // If an instance already exists, destroy this one
            Destroy(this.gameObject);
        }
    }

    Unity Message | 0 references
    void Start()
    {
        // Subscribe to the scene loading event
        SceneManager.sceneLoaded += OnSceneLoaded;
    }

    Unity Message | 0 references
    void OnDestroy()
    {
        // Unsubscribe from the scene loading event to prevent memory leaks
        SceneManager.sceneLoaded -= OnSceneLoaded;
    }

2 references
void OnSceneLoaded(Scene scene, LoadSceneMode mode)
{
    int sceneIndex = scene.buildIndex;

    // Reference to your AudioSource component
    AudioSource audioSource = GetComponent<AudioSource>();

    // Check the scene index and decide whether to play or stop the audio
    if (sceneIndex == 0 || sceneIndex == 1 || sceneIndex == 4) // Replace de
    {
        // Play audio if not already playing
        if (!audioSource.isPlaying)
        {
            audioSource.Play();
        }
    }
    else
    {
        // Stop the audio playback
        audioSource.Stop();
    }
}
```

```
public class LogicScript2 : MonoBehaviour
{
    public int playerScore;
    public TextMeshProUGUI scoreText;
    public GameObject gameOverScreen;
    public TextMeshProUGUI highscoreText;
    private int highscore = 0;
    private bool isGameOver2 = false; // Flag to track if the game is over.
    private bool isGameStarted2 = false; // Flag to track if the game has started.
    private const string HighscoreKey2 = "Highscore2";

    [ContextMenu("Increase Score")]
    1 reference
    public void addScore(int scoreToAdd)
    {
        if (isGameStarted2 && !isGameOver2)
        {
            playerScore += scoreToAdd;
            scoreText.text = playerScore.ToString();

            if (playerScore > highscore)
            {
                highscore = playerScore;
            }
        }
    }

    1 reference
    private void SaveHighscore()
    {
        PlayerPrefs.SetInt(HighscoreKey2, highscore);
        PlayerPrefs.Save();
    }
private void LoadHighscore()
{
    highscore = PlayerPrefs.GetInt(HighscoreKey2, 0);
}

Unity Message | 0 references
void Start()
{
    LoadHighscore(); // Initialize highscore from PlayerPrefs
}

// This function will be called when the button is clicked or space is pressed.
1 reference
public void restartGame()
{
    if (isGameOver2)
    {
        // Logic to restart the game.
        // For example, reloading the current scene:
        SceneManager.LoadScene(3);
        isGameOver2 = false; // Reset the game over flag.
        isGameStarted2 = false; // Reset the game started flag.
    }
}
```

```
void Update()
{
    // Check if the space key is pressed.
    if (Input.GetKeyDown(KeyCode.Space))
    {
        if (isGameStarted2)
        {
            // If space is pressed and the game is over, call the RestartGame function.
            restartGame();
        }
        else
        {
            // If the game hasn't started yet, set the game as started.
            isGameStarted2 = true;
        }
    }
}

0 references
public void returnToMenu()
{
    SceneManager.LoadScene(1);
}

2 references
public void gameOver2()
{
    gameOverScreen.SetActive(true);
    isGameOver2 = true;

    // Display the highscore when the game is over.
    highscoreText.text = "Highscore: " + highscore.ToString();

    SaveHighscore(); // Save the highscore when the game is over.
}
```

```
Unity Script (1 asset reference) | 0 references
public class ObstacleSpawnScript : MonoBehaviour
{
    public GameObject[] obstaclePrefabs; // Store your obstacle prefabs here
    public Transform spawnPoint;
    public float minSpawnDelay = 3.0f;
    public float maxSpawnDelay = 3.0f;

    private bool isSpawning = false;
    private bool isFirstObstacle = true;

    Unity Message | 0 references
    private void Start()
    {
        // Start the spawning coroutine
        StartCoroutine(SpawnObstacles());
    }

    1 reference
    private IEnumerator SpawnObstacles()
    {
        while (true)
        {
            if (!isSpawning)
            {
                isSpawning = true;

                if (isFirstObstacle)
                {
                    // Spawn the first obstacle immediately (excluding "Rocket 2")
                    SpawnFirstObstacle();
                    isFirstObstacle = false;
                }
                else
                {
                    // Wait for a random delay within the specified range
                    float delay = Random.Range(minSpawnDelay, maxSpawnDelay);
                    yield return new WaitForSeconds(delay);
```

```
                // Spawn the next obstacle
                SpawnObstacle();
            }

            isSpawning = false;
        }
        yield return null;
    }
}

1 reference
private void SpawnFirstObstacle()
{
    GameObject randomObstaclePrefab;

    // Keep reselecting until a non-"Rocket 2" obstacle is chosen
    do
    {
        randomObstaclePrefab = obstaclePrefabs[Random.Range(0, obstaclePrefabs.Length)];
    } while (randomObstaclePrefab.CompareTag("Rocket2"));

    // Get the position of the random obstacle prefab
    Vector3 spawnPosition = randomObstaclePrefab.transform.position;

    // Debug log to check which obstacle is spawned and its position
    Debug.Log("Spawning " + randomObstaclePrefab.name + " at position: " + spawnPosition);

    // Instantiate the obstacle at the original prefab's position
    GameObject spawnedObstacle = Instantiate(randomObstaclePrefab, spawnPosition, Quaternion.identity);

    // Ensure that the spawned obstacle has one of the specified tags
    if (spawnedObstacle.CompareTag("Spike1") || spawnedObstacle.CompareTag("Spike2") || spawnedObstacle.CompareTag("Rocket1") || spawnedObstacle.CompareTag("Rocket2"))
    {
        // Optionally, you can add more customization for the spawned obstacle here
    }
}

1 reference
private void SpawnObstacle()
{
    // Spawn a random obstacle from the array
    GameObject randomObstaclePrefab = obstaclePrefabs[Random.Range(0, obstaclePrefabs.Length)];

    // Get the position of the random obstacle prefab
    Vector3 spawnPosition = randomObstaclePrefab.transform.position;

    // Debug log to check which obstacle is spawned and its position
    Debug.Log("Spawning " + randomObstaclePrefab.name + " at position: " + spawnPosition);

    // Instantiate the obstacle at the original prefab's position
    GameObject spawnedObstacle = Instantiate(randomObstaclePrefab, spawnPosition, Quaternion.identity);

    // Ensure that the spawned obstacle has one of the specified tags
    if (spawnedObstacle.CompareTag("Spike1") || spawnedObstacle.CompareTag("Spike2") || spawnedObstacle.CompareTag("Rocket1") || spawnedObstacle.CompareTag("Rocket2"))
    {
        // Optionally, you can add more customization for the spawned obstacle here
    }
}
```

```
Unity Script (1 asset reference) | 0 references
public class RobotScript2 : MonoBehaviour
{
    public Rigidbody2D myRigidBody;
    public float jumpStrength;
    public LogicScript2 logic2;
    private bool robotIsAlive2 = true;

    // Reference to the CollideAudioScript component
    private CollideAudioScript collideAudio;
    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {
        logic2 = GameObject.FindGameObjectWithTag("Logic2").GetComponent<LogicScript2>();
        collideAudio = GameObject.FindGameObjectWithTag("CollideAudio").GetComponent<CollideAudioScript>();
    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Space) == true && robotIsAlive2)
        {
            myRigidBody.velocity = Vector2.up * jumpStrength;
        }

    }

    Unity Message | 0 references
    private void OnCollisionEnter2D(Collision2D collision)
    {
        // Check if the colliding object has the tag "Ground2"
        if (collision.gameObject.CompareTag("Ground2"))
        {
            // Do nothing or add any specific handling for "Ground2" collisions.
            // You can leave this block empty if you want to ignore collisions with "Ground2."
        }
        else
        {
            // Handle collisions with objects other than "Ground2"
            logic2.gameOver2();
            robotIsAlive2 = false;
            // Trigger the audio when a point is scored
            collideAudio.PlayCollideAudio();
            Destroy(gameObject);
            Debug.Log("Robot Deleted");
        }
    }

    //This method is called when the object becomes invisible
    Unity Message | 0 references
    private void OnBecameInvisible()
    {
        if (robotIsAlive2)
        {
            //Call the game over function when the character leaves the screen
            logic2.gameOver2();
            robotIsAlive2 = false;
            // Trigger the audio when a point is scored
            collideAudio.PlayCollideAudio();
            Destroy(gameObject);
            Debug.Log("Robot Deleted");
        }
    }
```