Универзитет “Св. Кирил и Методиј” – Скопје

Факултет за информатички науки и компјутерско инженерство

Дипломска работа

Изработка на видео игра во платформата за развој на игри со отворен код - Godot

Ментор:

Д-р. Катарина Тројачанец Динева

Изработил:

Иван Стојанов

Содржина

Апстракт

Целта на оваа дипломска работа е илустрирање на можностите на софтверската платформа за развој на игри Godot со создавање на видео игра од жанрот на одбрана на територии.

Дипломската работа ќе вклучи опис на основните концепти поврзани со развојот на видео игрите, како и платформите што овозможуваат создавање на игри, компонентите и алатките од кои тие се состојат, како и влијанието на жанровите видео игри врз развојот на ваквиот вид платформи. Посебен фокус во дипломската работа ќе се стави на основните карактеристики на платформата за развој на видео игри со отворен код Godot.

Дополнително, ќе биде разработено и создавањето на играта користејќи го интегрираниот програмски јазик GDScript и јазлите што ги нуди Godot од кои позначајни се Path2D, Tilemap, AnimationPlayer и најразличните Control јазли со кои ќе биде изграден корисничкиот интерфејс.

1. Вовед

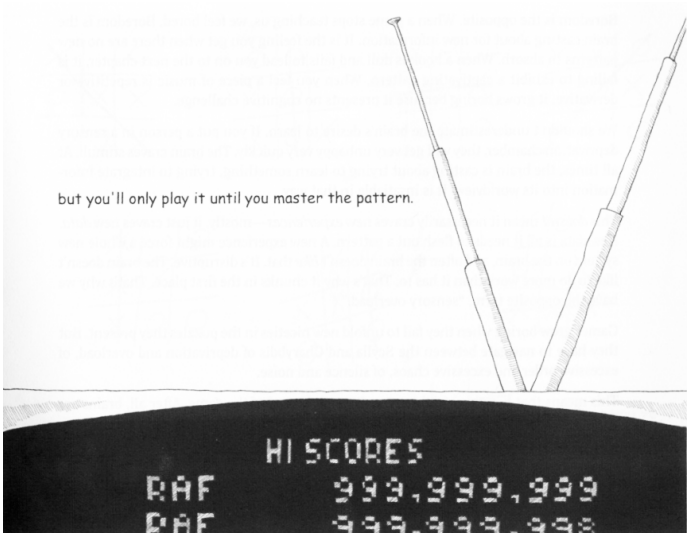
Поимот игра најчесто не носи кон интуитивно размислување за друштвени игри како што се шах, монопол, ризик или пак игри со карти како покер и блекџек, компјутерски игри, најразлични игри што ги играат деца заедно и безброј останати примери.

Во контекст на конзоли или пак, компјутерски базирана забава, зборот “игра” претставува приказ од повеќе слики кои потекнуваат од дводимензионален или тридимензионален виртуелен свет составен од хуманоиди, животни или возила како главен карактер што е под контрола на играчот.[1]

Главен фокус на оваа дипломска работа е создавање на видео игра во софтверската платформа со отворен код – Godot.[2]

1.1 Што е видео игра?

Видео игрите можеме да ги дефинираме како интерактивно искуство каде што играчот ќе биде соочен со секвенца од повторливи шеми на препреки. Играчот ќе има можност да ги надвладее, научи и да стане вешт (слика 1.1). Учењето и вештините се главниот причинител зошто ги нарекуваме игрите забавни, аналогно со тоа како шала ни станува смешна во моментот кога ќе ја сфатиме, препознавајќи некаква шема што се повторува. [3]



Слика 1.1 - Играта се игра сè додека играчот не стане вешт

Интеракцијата може да биде изведена преку допир на екран, џојстик, тастатура, глувче, контролер за виртуелна реалност и уште голем број на влезни уреди. Со наведените уреди за интеракција, може да се предизвикаат промени во виртуелниот свет во реално време.

Во однос на играта промените можат да бидат од тип на симулација, решавање на загатка или пак прецизно прескокнување на платформи. Дизајнот на една видео игра во голема мера се сведува

на психологија. Така на пример, ако се работи на „hyper casual“ игра, фокусот е ставен на делот како може да се направи играта зависна за играчот, а во исто време, механиките и играњето на играта да се сведат на ниво што е лесно сфатливо за секој.

Ако се разгледа дефиницијата за игра што беше напомената погоре, може да се забележи дека имаат заеднички елементи. Игрите најчесто се претставени како да се одвиваат во некој посебен свет. Најчесто се опишани како симулација, формален систем или, како што би рекол Хузинга, игрите претставуваат магичен круг, што нема допир со реалноста. [3]

Но, исто така, може да делува како тие да се поблиску до реалноста, колку и да сметаме дека се одвиваат во виртуелен свет. Причината за ова е во тоа што игрите се апстрахирани од реалноста, односно, се преточени од шеми што ги гледаме во реалниот свет. Самите игри имаат поголема поврзаност со реалниот свет од што првично се добива впечаток. Споредбено со тоа како ние ја доживуваме реалноста што претставува наша лична апстракција односно перспектива како го гледаме светот.[4]

1.2 Што е платформа за развој на игри?

Платформа за развој на игри е софтверско решение што е дизајнирано за создавање на видео игри. Првата платформа се појавила на средина на деведесетите како резултат на создавање на игра со пукање во прво лице наречена Doom од компанијата ID Software.[5]

Doom била структурирана со добро дефинирана одвоеност меѓу главните софтверски компоненти како: тридимензионален графичка рамка за рендерирање, детекција на колизија или пак аудио рамка и самите графички ресурси кои се користеле, световите и правилата за играње кои се составен дел од искуството на играчот.

Важноста на ваквата поделба станала видлива во моментот кога компаниите за развивање на видео игри почнале да ги лиценцираат своите игри и со постоечките алатки да создаваат нови производи кои се разликувале во графичките ресурси, поставеноста на светот, оружјата, карактерите, возилата и нови правила кои претставувале само минимална промена на веќе постоечката платформа. Овој период го бележи почетокот на „modding“ заедницата [24] која се состои од група на индивидуални играчи на игри или мали независни студија кои граделе нови игри базирани на веќе постоечка игра со помош на алатки што биле направени достапни од оригиналните развивачи.

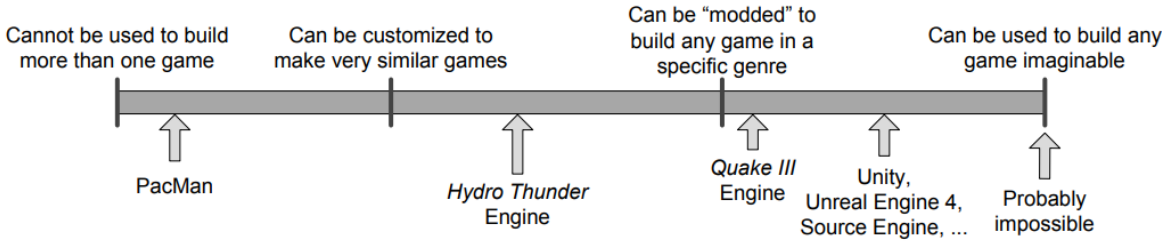
Кон крајот на деведесеттите, некои игри како Quake III Arena и Unreal биле дизајнирани со цел да бидат реискористени. Модификација на платформата станало возможно со помош на сложените програмските јазици како QuakeC.[6]

Лиценцирањето на платформите за создавање на игри станал втор начин на заработка за тие што развиваат видео игри. Денес тие што изработуваат видео игри можат да лиценцираат платформа и да реискористат клучни функционалности кои им се потребни за создавање на нивната игра. Практиката да се користат платформи за создавање на видео игри наместо да се создава слично решение се покажало економски исплатливо.

Границата помеѓу игра и платформа е бледа. Некои платформи за развој на игри драстично се флексибилни и се истакнуваат како моќна алатка за развој на видео игри, додека пак други, не можат да направат остра граница меѓу двете. Добар пример е RPG Game Maker [7], што е наменет исклучиво за создавање на 2Д игри од птичја перспектива и истата платформа би поставила непотребен предизвик да се користи за развивање на 3Д или пак игра што не спаѓа во специфичен жанр. Иако развивањето е возможно, целта на платформата е да биде корисна алатка што ќе ни го олесни процесот на развивање со промена на фокусот кон играта, а не алатката што ја создала.

Не постои гејм студио што јасно прави поделба меѓу игра и платформа, што е до некаде очекувано земајќи во предвид дека овие две компоненти често се променливи, сè додека дизајнот на играта не е цврсто дефиниран.[1]

Архитектура управувана од податоци овозможува де се направи разлика меѓу софтверска платформа за развој на видео игри и кое било друго софтверско решение. Кога играта има нефлексибилна логика и правила или содржи посебен код за рендерирање на одреден тип на објекти станува тешко или невозможно повторно да се искористи софтверот за да се направи различна игра. Терминот “платформа за развој на игри” се однесува на софтвер што е флексибилен и овозможува да се користи како основа за создавање на повеќе различни видови на игри, без потреба од поголеми модификации.



Слика 1.2 - Историја на реискористливост на платформите за развој на видео игри

2. Платформа за развој на видео игри

Платформите за развој на видео игри можат да се категоризираат како платформи за општа употреба и специјализирани платформи. Тие кои спаѓаат во категоријата за општа употреба треба да се флексибилни и да поседуваат алатки кои би се користеле за најразличен вид видео игри, без сериозни ограничувања за одреден жанр или вид на игра. Godot, Unreal Engine, Unity, CryEngine спаѓаат во категоријата на платформи за развој на игри за општа употреба.

Од друга страна, специјализираните платформи се наменети за создавање на конкретен тип на игри, со цел задоволување на потребите за перформанси или пак специфични алатки кои се потребни за развивачите на истата. Добар пример за ваков вид платформи е Serious Engine [9] што е користен од Хрватската компанија Croteam за изработка на франшизата Serious Sam. Оваа платформа се карактеризира со тоа што може да прикаже голем број на непријатели и притоа да рендерира модерни визуелни ефекти, големи текстури и динамичко осветлување. Благодарение на добрата структура на платформата, успева да ги постигне резултатите со минимална загуба на визуелниот квалитет.

Ако се продлабочи поимот платформа за развој на видео игри и нејзината дефиниција, може да се сфати како компјутер што извршува процеси. Компјутер на најниско ниво се состои од кола кои имаат две состојби единица или нула. Кога голем број на вакви вредности ќе ги соединиме во функции го добиваме оперативниот систем што потоа може да извршува апликации создадени од сложени програмски јазици. Овие програмски јазици се користат за создавање на нашите омилени веб прелистувачи и платформи за развој на игри. Компјутерите работат на принцип што може да биде опишан како слоеви на апстрахирана комплексност. Платформите лежат на врвот на скалата на горенаведената комплексност. [1]

Со една реченица платформа за развој на игри ја дефинираме како софтверска рамка, дизајнирана за создавање и развивање на видео игри.

2.1 Компоненти на платформа за развој на игри

Платформа за развој на игри се состои од множество на алатки и извршна компонента. Во продолжение ќе наброиме неколку алатки и потсистеми на извршната компонента кои се основен составен дел од повеќето платформи.

Основни компоненти на една платформа претставуваат:

- Рендерирање (2Д и 3Д): Рендерирање е процес каде што видео играта ја исцртуваме на екранот на корисникот. Добра процедура за рендерирање (анг. Rendering pipeline) зема во предвид поддршка за модерните графички картички, дисплеи со висока резолуција и ефекти како осветлување, перспектива и поглед (анг. Viewport), притоа одржувајќи доволно висок број на слики во секунда (анг. Frames per second).
- Физика и колизија: Систем што е составен дел на речиси секоја платформа за развој на игри. Градење на робусен и прецизен систем за физика е неопходно. Повеќето игри имаат

потреба од детекција и одговор на колизија, реалистична симулација на физика, но многу ретко развивачите на игри или софтвер би сакале во свои раце да ја превземат одговорноста за изработка на ваквите системи.

- Поддршка за повеќе платформи: Развивачите на видео игри имаат потреба да ги издадат своите дела на повеќе платформи како конзоли, паметни телефони, персонални компјутери или пак на веб прелистувачи. Една модерна платформа нуди унифициран процес на експортирање за повеќе платформи без потребата да се развива играта од почеток за одредена платформа. Се разбира треба да се земат предвид уредите за влез, излез и перформансите што ги поседува платформата. Основен пример би било паметен телефон наспроти персонален компјутер каде предизвикот е користење на допир наместо тастатура и глувче.
- Заедничка работна околина: Со користење на истиот интерфејс и начин на работа, возможно е да се развива секаков тип на игра без потреба да се учи нов начин на работа за секој проект.

Дополнително, модерната платформа се состои од множество алатки кои помагаат при развивање на играта и вклучување различни опции, како олеснување на процесот на работа со слики или звуци, анимации, дебагирање, креирање нивоа, можност играта да се игра во мрежа и многу други алатки. Често платформите за развој на игри имаат вградена алатка што помага со вклучување на содржина како анимации или 3Д модели од друг софтвер.

2.1.1 Потсистеми на извршна компонента

Како и многу други софтверски системи, платформите имаат архитектура во повеќе нивоа. Во општ случај, потсистемите од повисоките нивоа зависат од потсистемите од пониските нивоа, но спротивното не важи. Како пример можеме да ги земеме менијата во видео игрите или кој било кориснички интерфејс што се наоѓа на повисоко ниво и е зависен од подсистемот за графика што се наоѓа на пониско ниво.

2.1.1.1 Графика

Потсистем кои нуди функционалности за работа со 2Д и 3Д графика и претставува еден од најсложените потсистеми. Повеќето рамки за рендерирање се изградени врз една од наведените библиотеки кои овозможуваат комуникација со хардверот на ниско ниво, и тоа се:

- *Glide* е 3Д графичка библиотека развиена од 3dfx Interactive наменета за старите Voodoo графички картички. Оваа библиотека станала популарна во ерата на хардверски T&L. T&L е термин што се користи во компјутерската графика за трансформација, клипинг и осветлување. Трансформација е процесот на создавање дводимензионален поглед на сцена што била во три димензии. Клипинг претставува процес на исцртување на сцена што ќе биде присутна во сликата откако рендерирањето завршило. Осветлување претставува задача на промена на боите соодветно на информациите за осветлување кои ги поседува објектот[10]

- *OpenGL* (Open Graphics Library) претставува стандардна спецификација на ниско ниво што се користи на повеќе платформи за пишување апликации кои создаваат 2D и 3D компјутерска графика
- *DirectX* е 3Д графички софтверски кит (ang. Software Development Kit) за развој на Microsoft и е главен ривал на OpenGL
- *Vulkan* развиен од AMD претставува наследникот на OpenGL и е наменет да нуди супериорни перформанси и поефикасна искористливост на централната обработувачка единица и графичкиот процесор во споредба со OpenGL и DirectX 11
- *Libgcm* претставува рамка на ниско ниво развиена од страна на Sony за хардверот на конзолата PlayStation 3 како поефикасна алтернатива на OpenGL
- *Edge* е моќен и високо ефикасна рамка за рендерирање и анимација создадед од Naughty Dog и Sony за PlayStation 3, што бил користен од многу трети развивачи на видео игри за истата платформа.

Како споредба платформата што ние ќе ја користиме Godot користи OpenGL ES 2.0 за компатибилност со постари уреди, а на поновите уреди со можност за користење на OpenGL ES 3.0. Според планот на развивање за платформата, до крајот на 2022 година ќе има вградено поддршка за Vulkan.

2.1.1.2 Физика и колизија

Компонента одговорна за физичката симулација на светот и детектирање дали објектите се во меѓусебен судир. Голем дел од популарните платформи користат софтвер за физика од трети компании како што се:

- *Havok* - софтвер докажан во индустријата, се користи во огромен број на игри развивани од големи студија
- *PhysX* - софтвер развиен од Nvidia што претставува популарен избор кај големите студија
- *Bullet* - софтвер со отворен код, често користен во научни области.

Платформата за развој на игри Godot што ќе ја користам со цел практична изработка на игра, има интегрирана библиотека за физика наречена Godot Physics, но постои и официјален пакет што овозможува поддршка за библиотеката со отворен код Bullet.

2.1.1.3 Менаџер на ресурси

Менаџер на ресурси е потсистем што е вграден во секоја платформа. Тој нуди унифициран интрерфејс преку што ни се достапни текстури, видеа, модели, скрипти и останати потребни ресурси. Повеќето платформи за развој на игри имаат централизиран и конзистентен пристап, додека останати платформи користат ад хок приод најчесто оставајќи на програмерот директно да ги вчитува необработените фајлови, или пак во компресирани датотеки како што е случајот со PAK фајловите на Quake платформата.

2.1.1.4 Звук

Аудио е еквивалентно битно како графика во платформите за развој на игри. За жал најчесто добива помалку внимание отколку рендерирање, физика, анимација, вештачка интелигенција и гејмплеј. Интересен факт е тоа што програмерите најчесто пишуваат код со изгасени звучници. Во секој случај ниту една игра не е комплетна ако нема неверојатна аудио рамка.

Аудио рамката на Quake е прилично едноставен и развивачите на игри најчесто додаваат сопствен слој на функционалност врз него, или пак создаваат свои софтверски решенија.

Unreal Engine 4 нуди робусна 3Д аудио рамка, иако со бројот на карактеристики делува лимитиран. Развивачите на видео игри најчесто се надоврзуваат на истиот и сакаат да го приспособат за сопствени цели, или цели на играта.

За DirectX платформи (персонални компјутери, Xbox 360, Xbox One, Xbox Series S/X) Microsoft нуди одличен пакет наречен XACT, подржан во извршување од XAudio2 и X3DAudio апликациски програмски интерфејс (анг. Application Programming Interface). Electronic Arts имаат развиено напреден внатрешен аудио рамка наречена SoundR!ot.

Од друга страна студија како Naughty Dog, Sony Computer Entertainment America (SCEA) нуди моќен 3Д аудио рамка наречена Scream, што бил користен од голем број на PlayStation 3 наслови вклучувајќи Uncharted 3 и The Last of Us. [1]

Аудио рамката исто како сите останати компоненти во една игра има потреба од посебен софтверски развој, интеграција, внимателно подесување и посветување внимание на деталите со цел да се постигне аудио од висок квалитет во финалниот продукт.

2.1.1.5 Систем за скриптирање

Многу платформи за развој на игри нудат јазик за скриптирање на високо ниво со цел да го направат развивањето на правилата на играта побрзо и полесно. Без постоењето на скриптирачки јазик би морале при секоја промена на логиката целосно да ја компајлираме нашата игра. Но кога јазикот за скриптирање е интегриран во платформата, промените во логиката или податоците можат да бидат направени само со модифицирање на скриптата, вчитување и компајлирање само на таа единечна скрипта. Некои платформите овозможуваат промените во скриптата да бидат вчитани додека играта се извршува. Други платформи пак мораат да бидат изгасени со цел скриптите да бидат компајлирани. Но како и да е, времето е драстично намалено со тоа што немаме потреба да чекаме да се компајлира целосната игра.

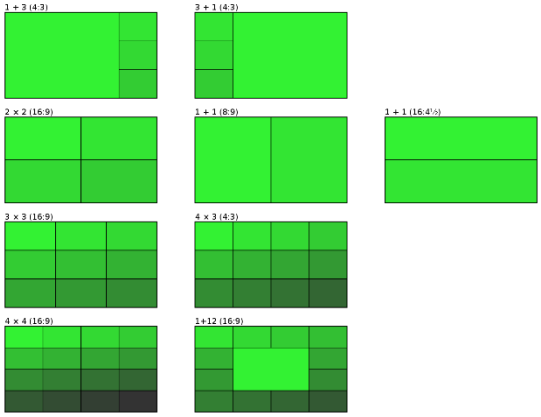
2.1.1.6 Мрежно играње

Многу игри дозволуваат повеќе играчи да се најдат во еден виртуелен свет, тоа не насочува кон игрите во мрежа.

Според дефиниција постојат четири начини како повеќе играчи можат да се сретнат во виртуелен свет:

- *Повеќе играчи на еден екран* – Два или повеќе уреди за внес се поврзани за аркадна машина, конзола или персонален компјутер. Повеќе играчи населуваат еден виртуелен свет со користење на единечна камера што ги прикажува сите играчи паралелно. Пример за вакви игри се Smash Brothers, Lego Star Wars и Gauntlet.

- *Повеќе играчи со одвоен екран* - Најчесто два или повеќе играчи населуваат еден виртуелен свет, со повеќе влезни уреди приклучени на една машина на што се извршува еден виртуелен свет. Но, во споредба со останатите, иако играчите играат на ист екран тие имаат посебни камери. Екранот е поделен во секции така што секој играч има сопствена секција каде што може да го следи својот карактер. На сликата подолу (слика 2.1) може да се видат најразлични поделби на екранот. Најчесто користени се 1+1 со однос 8:9 (вертикална поставеност на секциите) и 1+1 со однос 16:4½(хоризонтална поставеност на секциите), најоптимални кога се работи за два играчи, 2x2 со однос 16:9 ,најоптимален кога станува збор за 4 играчи. Со овие секции екранот бил максимално искористен во однос на бројот на играчи кои играле во исто време. Пример на екран поделен на секции од играта Crash Team Racing – 1999 е даден на слика 2.1.



Слика 2.1 - Поделба на екран во секции

- *Мрежно играње со повеќе играчи*. Повеќе компјутери или конзоли се поврзани заедно во мрежа, додека еден од играчите е хост или пак играта е хостирана на сервер со посебна намена.
- *Масивен број на играчи поврзани паралелно во еден виртуелен свет* (анг. Massive Multi-player Online Game – MMOG). Подржува стотици до илјадници играчи поврзани во исто време на еден гигантски свет што е хостиран на посебен сервер.

Првите два начини за играчите да се сретнат во виртуелниот свет биле создадени поради тоа што интернетот сеуште имал висока цена или не постоел во одредени региони, означувајќи го почетокот на игрите со повеќе играчи.

Слика 2.2 - Пример за екран поделен на секции од играта Crash Team Racing – 1999 [11]

Игрите за повеќе играчи се многу слични на игрите наменети за еден играч. Иако поддршка на повеќе играчи може да има значително влијание на дизајнот на одредени компоненти на софтверската платформа за развој на игри. Објектот на светот во играта, рендерот, влезните уреди, системот за контрола на играчот и системите за анимација се афектирани. Приспособување на игра наменета за еден играч во игра за повеќе играчи иако не е невозможно претставува огромен предизвик. Таков предизвик преземале многу од студијата за развивање на игри. Тие ги пренамениле своите игри во игри за повеќе играчи. Но сепак е препорачливо ако играта сакаме да биде за повеќе играчи истото да биде дизајнирано од самиот почеток со цел да се намалат трошоците во иднина.

Додека пак мигрирање на игра за повеќе играчи во игра за еден се покажало тривијално. Многу платформи за развој на игри ги третираат игрите за еден играч како инстанца од игра за повеќе играчи, со цел овозможување полесна миграција ако се донесе таква одлука. Quake платформата е познат по client-on-top-of-server mode во кој се компајлира една извршна програма што се извршува на само еден персонален компјутер и истата таа програма делува како клиент и како сервер.

2.1.2 Алатки

За изработка на една видео игра, потребни се голем број на ресурси. Во нив спаѓаат уметнички, конфигурациски датотеки, скрипти и слично. Алатките ја олеснуваат работата на развивачите на игри и уметниците. Тие не се дел од извршната верзија на играта. Во зависност од играта, програмерите можат да развијат алатки со специјална намена за истата. Се разбира постојат алатки кои се заеднички и се користат за развивање на различни типови на игри како:

- Уредувач на свет – интерактивна алатка што претставува главен прозорец во повеќето платформи преку што се врши уредување на виртуелниот свет на играта. Преку оваа алатка се дизајнира светот и овозможува транслација, ротација и скалирање.
- Инспектор – основна алатка на секоја платформа за развој на игри што овозможува разгледување на карактеристиките што ги има еден објект. Исто така во алатката имаме опција да иницијализираме променливи на нашите скрипти или пак да подесуваме променливи како на пример ширина, висина, обојување на слика и базичните променливи кои се однесуваат на трансформација на објектот.
- Конвертер на ресурси – ресурсите кои се создадени од артистите најчесто не се во соодветен формат за платформата. Најчесто користени формати за слики се JPEG, PNG и SVG, а за аудио се користат WAV, MP3 и Ogg, за 3Д моделите стандардизирани формати се obj и fbx кои се наменети да бидат универзални, но не секогаш се добиваат добри перформанси доколку директно се користат. Затоа, повеќето платформи воведуваат формати со кои го претставуваат светот и објектите во него.
- Тајлмап едитор – претставува алатка што овозможува “боење” на нивоа или мапа, клучна алатка што се користи во повеќето дводимензионални игри. Поголема слика се дели на квадрати со одредена димензија (пример: 64x64 пиксели). Потоа со дадените исечоци создаваме безброј различни средини со претходно создадената палета.

- Продавница за ресурси – во Unity, Unreal и Godot ако ви се потребни ресурси за создавање на видео игра можете многу лесно да ги најдете во нивните продавници за ресурси. Ресурсите во Unity се зачувуваат во датотека unitypackage, што претставува датотека што платформата може да ја вчита и истиот го олеснува процесот на имплементација. Во продавницата возможно е да се преземат текстури, слики од дводимензионални карактери, 3Д модели или пак музика и аудио ефекти кои би го потпомогнале процесот на создавање на игра. Исто така возможно е да се најдат алатки за платформата, шаблон на целосна видео игра или пак одреден систем/механика што може да се имплементира и модуларно да се користи во повеќе игри.
- Систем за контрола на верзии – алатка што се користи при развивање на софтвер особено кога се работи за проект на кој работат повеќе инженери. Најчесто користена алатка е Git, но платформа како Unity нуди вградена алатка наречена Unity Collaborate. Оваа алатка ни овозможува централизирано поставување на проектот на еден сервер. Секоја промена на проектот се запишува, промените можат да ги преземат сите што се дел од проектот и овозможува враќање на верзии. Дополнителна опција е создавање на посебна гранка со цел развивање на нов потсистем што понатаму ќе се имплементира во главниот проект.
- Систем за генерирање на честички – претставува корисна алатка за создавање на богати визуелни ефекти. Алатката е комплексна и се состои од голем број на параметри кои помагаат за изработка на посакуваниот ефект.
- Алатка за создавање на анимации – составен дел од секоја модерна платформа и овозможува создавање на анимации на 2Д и 3Д објекти во самата платформа.
- Уредувач на скрипти – уредувачите на скрипти најчесто се екстерни како на пример Visual Studio Code, Visual Studio и MonoDevelop. Во случајот на Godot почетната поставка е интегрираниот уредувач на текст што нуди дебагирање, подвлекување на синтаксички грешки, интелигентно дополнување на код и кратенки.

Развивањето на ваквите алатки е долг и скап процес. Unity или пак Godot доаѓаат со пакет од алатки, но сепак овозможуваат развивање на специјализирани алатки кои ќе го помогнат развивањето на играта. Во рамки на оваа дипломска работа во софтверската платформа Godot ќе бидат искористени горенаведените алатки.

2.2 Развојот на платформите во однос на жанровите

Платформите за развој на игри можат да бидат специјализирани за создавање на игри од одреден жанр. Платформа што била направена за игра со борење во ринг е драстично поразлична од игра со масивен број на играчи поврзани мрежно или пак од стратегиска игра во реално време.

Игрите од различните жанрови сепак имаат нешто заедничко, на пример ако игрите се изведени во три димензионален простор без разлика на жанрот нив ќе им биде потребен еден или друг тип на влез за интеракција со виртуелниот свет, потреба од 3Д меш рендеринг (анг. 3D mesh rendering), некаков тип на худ (анг. Heads-up Display) што вклучува рендерирање на текст и широк избор на фонтови, аудио рамка и останати компоненти кои се составен дел на една видео игра.

Unreal Engine иако првично бил наменет за создавање на игри со пукање во прво лице денес успешно се користи за создавање на игри од голем број на различни жанрови.[8]

Игрите со **пукање во прво лице** (ang. First Person Shooter) претставуваат жанр што се истакнува со игри како Quake, Unreal Tournament, Half-Life, Counter-Strike и Battlefield. Овој тип на игри историски вклучувале систем за движење во потенцијално голем свет составен од коридори. Модерните игри со пукање во прво лице се состојат од добро структурирани околини кои се составени од широко отворени области на поле и строго затворени области. Исто така се карактеризираат со маханики за движење или транспорт во кои се вклучуваат локомоторен систем, шински транспорт или возила, авиони, бродови и различни типови на летала.

Игрите од овој жанр се типично најтешки за развивање од технолошка перспектива. Тежината потекнува од тоа што овој тип на игри создаваат илузија на свет со многу детали и хиперреализам за играчот. Најголемите технолошки иновации во индустријата за видео игри потекнува токму од овој жанр.

Технологиите кои се во фокус на игрите со пукање во прво лице:

- Ефективно рендерирање на масивен 3Д виртуелен свет
- Осетлива камера и махиники за нишанење
- Реалистични анимации на рацете и оружјата на играчот
- Разновидност на оружја кои треба да се верни на тие во реалноста
- Опростувачко движење и модел на колизија, што најчесто ги прави играчите да се осеќаат како да лебдат
- Реалистични анимации и вештачка интелигенција на карактерите кои не се контролирани од играч (анг. Non-player character – NPCs)
- Можност за играње во мрежа, најчесто со поддршка до 64 играчи во исто време

Игрите со борење се типично наменети за двајца играчи што вклучуваат хуманоид карактери кои се борат во некаков тип на ринг. Овој жанр го претставуваат игрите како Soul Calibur, Tekken, Mortal Kombat и Street Fighter. [12]

Традиционални игри во боречкиот жанр се фокусираат кон технолошкиот развој на:

- Богати боречки анимации
- Прецизна детекција на удар
- Детекција на кориснички влез што се состои од комплексни комбинации на копчиња
- Публика, статичка позадина и динамичен терен што се менува во однос на кулминацијата на борбата

Модерните **стратегиски игри во реално време** (анг. Real-time Strategy) претставуваат жанр на игри кои биле дефинирани од игрите како Dune II: The Building of a Dynasty (1992). Останати игри кои спаѓаат во истиот жанр се Warcraft, Command & Conquer, Age of Empires и Starcraft.

Во овој жанр играчот организира војници низ големо поле со цел да ги надигра своите непријатели. Карактеристично за овие игри е светот да биде прикажан од птичја перспектива со ограничен поглед врз територииите што ги поседува или откриените територии. Се останато е

претставено со “црна магла” што не дозволува да погледне играчот што се случува во останатите предели.

Постарите стратегиски игри користеле мрежест дизајн со поделба на ќелии каде што било овозможено градење и била користена ортографска проекција со цел да се поедностави рендерирањето.

Слика 2.3 - Age of Empires создадена од Ensemble Studios

На Слика 2.3 е прикажано како изгледа екран од класичната RTS игра Age of Empires. Модерни стратегиски игри некој пат користат перспективна проекција и вистински три димензионален свет, но повторно може да се состојат од поделба репрезентирана во ќелии со цел војниците и позадинските елементи како градби да бидат меѓусебно порамнети.

Некои други слични карактеристики што ги имаат стратегиските игри во реално време вклучуваат:

- Секој војник во играта е рендериран во низок квалитет со цел да може играта да подржи голем број на војници во исто време
- Нерамен терен е најчесто платното врз кое овој тип на игри се играат и дизајнираат
- На играчот најчесто му е дозволено да гради нови градби кои му овозможуваат создавање на одреден тип на војници

- Корисничката интеракција најчесто се изведува со единечен клик за контрола на еден војник, селекција на одреден регион со цел контролирање на повеќе војници, мениа или лентата со алатки кои се состојат од команди, опрема, типови на војници, типови на градби итн.

Слика 2.4 - Northgard модерна RTS игра

На слика 2.4 е прикажана модерна стратегиска игра во реално време наречена Northgard [13] што е создадена од студиото Shiro Games со користење на платформата Heaps.io[14]. Heaps.io платформата е бесплатна и со отворен код што овозможува лесно модифицирање. Главната карактеристика што ја истакнува платформата е испорачување на брзи итерации при развивање, поддршка за повеќе платформи и минимална цена во однос на перформанси. Исто така платформата овозможува прилагодливи процедури за рендерирање во однос на графичките потреби кои ги имаат развивачите на игри. Northgard е една од најпознатата игра направена во оваа платформа.

Како што претходно беше напоменато, Northgard е стратегиска игра што користи перспективна камера, теренот е претставен со хексоаголни секции кои се состојат од нерамнини, иглолисни дрва, езера и камења кои означуваат разновидни ресурси. На секциите кои ги поседува играчот му е овозможено да гради. Тие градби можат да имаат најразлични намени како производство на војска, куќичка изградена во близина на езеро во кои живеат работници кои се занимаваат со рибарење, дрвосечач или ковач кој ги подобрува орудијата. Градбите кои може да се изградат се тесно поврзани со теренот што е поседуван.

Може да се дојде до заклучок дека секој жанр на игри има посебни технички барања кои треба да бидат задоволени. Следствено стигнуваме до објаснувањето зошто платформите се разликувале од жанр до жанр. Но исто така може да се види дека помеѓу различните жанрови има големо преклопување, особено кога се зборува за игри кои потекнуваат од иста хардверска платформа.

Како што хардверот станува се помоќен стигнуваме до заклучок дека разликите помеѓу хардверските платформи кои потекнувале од хардверско ограничување и соодветна оптимизација за специфичните игри полека исчезнува. Поради истото се појавува можност на развивачите на видео игри да користат една платформа за развој на игри за повеќе жанрови и притоа да се елиминира сериозната пречка за издавање на повеќе платформи.

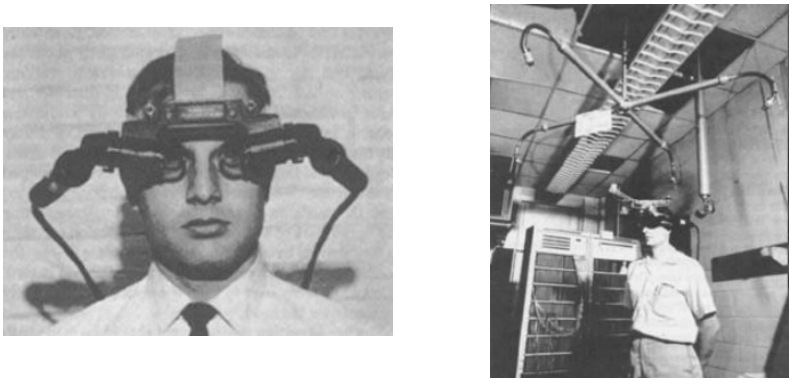
2.3 Поширока примена на платформите за развој на игри

Границите помеѓу виртуелниот и физичкиот свет почнуваме да ги стеснуваме и причинителот за тоа е технолошкиот напредок што сме го имале низ годините предизвикан од индустријата за видео игри. Microsoft Flight Simulator е одличен пример за истото. Тој ни овозможува да летаме врз свет што е верна реплика на нашата планета. Игрите на Sony прикажуваат високо реалистични фацијални анимации и Half-Life: Alyx создадена од страна на Valve постави нов стандард за како едно виртуелно искуство треба да изгледа.

На страна од индустријата за видео игри, платформите започнуваат да се користат за развивање на софтвер што не е видео игра. Примените можеме да ги најдеме во филмската индустрија, автомобилската индустрија, виртуелната и надградената реалност.

2.3.1 Виртуелна реалност (анг. Virtual Reality - VR)

Виртуелна реалност генерално може да биде дефинирана како виртуелен објект во виртуелна околина, или поточно како прецизна симулација или вештачки дупликат на реалниот свет што е компјутерски генериран. Давајќи му на корисникот искуство од симулирана реалност со манипулирање примарно на сетилата за слух и вид. Одредени уреди за виртуелна реалност вклучуваат и манипулација на сетилата за допир и мирис.



Слика 2.5 - Ivan Sutherland креаторот на првиот визир за виртуелна реалност наречен “The Sword of Damocles” (1968)

Поимот виртуелно означува ефект што не е реалноста. Реалноста ја дефинираме како нешто што е реално или ефективно отколку нешто што е очигледно. Додека пак виртуелната реалност е дефинирана како околина што е генерирана од компјутерски системи каде што корисникот може да се однесува како да е реална или наједноставно опишано како виртуелен свет што постои само во компјутерите и умот.

Добар систем за виртуелна реалност би им дозволил на корисниците да навигираат околу физички објекти и да имаат контакт со истите како да припаѓаат на реалниот свет. Постепено би

можеле виртуелната реалност да ја класифицираме во два различни вида и тоа: невнесувачка (анг. non-immersive) и внесувачка (анг. immersive). Невнесувачките би претставувале компјутерска симулација од реалниот свет, додека пак внесувачките би ги имале истите карактеристики како и невнесувачките само што корисникот би се чувствувал како да е целосно извлечен од околината во која се наоѓа користејќи визир. [17]

2.3.2 Надградена реалност (анг. Augmented Reality - AR)

Терминот надградена реалност првпат се спомнува во Boeing во 1990 од истражувачот Tom Caudell кој бил одговорен за развивањето на опрема за означување дијаграми и уреди кои ги насочувале вработените низ централата. Тој предложил да ги заменат дрвените панели кои се состоеле од инструкции за поврзување на електрониката индивидуално за секое летало со визир што би можел да ги прикаже потребните скици во однос на леталото. Тој визир би претставувал технолошки напредни очила кои би проектирале на реискористливи панели. Од тогаш надградената реалност се користи да се подобри корисничкото искуство во различни задачи со додавање на ниво на компјутерски генерирани надградби на веќе постоечката реалност со цел да ја направи по значајна поради тоа што можеме да имаме интеракција со истата. Со други зборови техниката на надградената реалност е комбинација од реалниот свет и компјутерски генерирана слика.

Корисникот може да го гледа реалниот свет во реално време, но со додадени виртуелни објекти. Овие виртуелни објекти се интегрирани во светот на корисникот користејќи преносни уреди. Технологијата за интеракција во реално време придонесува корисникот да се чувствува како виртуелните објекти да не се разликуваат од објектите во реалниот свет. Како пример, корисникот може да види маса на која има вистинска чаша со вода и компјутерски генерирана чаша со вода. Најбитниот аспект од надградената реалност е тоа што локацијата на виртуелната чаша се прикажува како вистинита и веродостојна во однос на физичката чаша.

Слика 2.6 - Популарната игра Pokemon Go што користи надградена реалност

Се повеќе истражувачи се обидуваат да се надоврзат на терминот надградена реалност како повеќе од систем што е комплимент на реалниот свет со компјутерски генерирани објекти. Тие сметаат дека областа е сеуште во почетните фази на развивањето на интеракциите кои се можни.

Со зголемениот број на мобилни уреди со можност да генерираат феноменални околини се очекува надградената реалност понатамошно да се развива.[17]

2.3.3 Мешана Реалност (анг. Mixed Reality – MR)

Мешаната реалност е дефинирана како хибридна реалност, каде реалниот и виртуелниот свет се комбинираат да создадат нови околини и визуелизации кои коегзистираат и имаат интеракции во реално време со физички и дигитални објекти. Хибридната реалност не се одвива само во физичкиот или само во виртуелниот свет. Односно претставува комбинација од виртуелна и надградена реалност. Мешаната реалност има можност да ги комбинира рендерираните објекти во реална околина, наведената техника се нарекува холографија.

Во реални и виртуелни околини со флексибилност, внесеност, интеракција, коегзистирање и збогатување се суштински карактеристики на едно искуство од мешаната реалност. Искуството се постигнува со технолошките карактеристики од надградена и виртуелна реалност. Така што хибридната реалност нуди реално виртуелна околина каде што корисниците се чуствуваат како да се обземени во околината и нивната перцепција на реалниот свет е подобрена.

2.3.4 Филмската индустрија

Филмовите и сериите во последната декада почнале да користат компјутерски генерирани слики кои се сметаат за неопходни на еден модерен филм. Тимот одговорен за создавање на серијалот Mandalorian (слика 2.7) го започнале иновативниот пат за продуцирање на филмови со користење на платформа за развој на игри. Тие ја користеле Unreal платформата за прикажување на околината во која актерите се наоѓале. Тоа придонесува сцената да биде променета во реално време со цел да биде адекватна на кадарот што се снима.

Слика 2.7 - Виртуелна продукција на сцена од серијалот Mandalorian[15]

Фирмата што е одговорна за визуелните ефекти на Star Wars, Industrial Light & Magic со видео прикажале како они ја користат платформата на Epic Games со цел креирање на прекрасните

пејзажи кои се наоѓаат во серијалот. Тие изгледаат феноменално и претставуваат зачеток на масивна револуција во филмската индустрија со елиминирање на зелената позадина и користење на виртуелна позадина што се извршува во реално време.

Компанијата ја користела Unreal платформата за градење на 3Д околина за Star Wars серијалот, притоа прикажувајќи ја на LED екран што ја обвиткал сцената. Ова значело дека се би можело да биде опфатено од камерата, но исто така околината во која се наоѓале актерите им помагала да ги подобрат своите перформанси поради тоа што наместо да гледаат во зелена позадина тие сега можеле да се соживеат со пејзаж од пустина.

Пред неколку декади дизајнери ги цртале пејзажите рачно, надевајќи се дека никој не би приметил дека гранките на дрвјата или пак облаците не се помрднуваат. Сега со оваа иновација би имале можност да изградат анимирана околина со можност за вметнување на анимирани карактери и притоа нивното поставување би можело да биде променето при самото снимање.[15]

2.3.5 Автомобилска индустријата

Надвор од индустријата за забава, Unity платформата се користи од страна на производителот на коли Volvo (слика 2.8) со цел да ја зголеми безбедноста на своите возила, поедноставувајќи го процесот на дизајнирање и како алатка за машинско учење.

Слика 2.8 - Намената на Unity во автомобилската индустрија [16]

Приоритет на возилата на Volvo од секогаш била безбедноста и они се првите кои сакаат да направат еден чекор понапред во тој аспект.

Со помош на Unity безбедноста на возилата може да биде тестирана во виртуелен свет со многу различни сценарија како што е прикажано во видеото погоре. Како пример имаме инстанцата кога корисникот го вози возилото и му е поставена пречка лос кој го преминува патот. Различните сценарија им овозможува да ги тестираат нивните возила и притоа да се истакнат грешките кои не би сакале во реалниот свет да се случат.

Примената на Unity во автомобилската индустрија од страна на Volvo продолжува со тестирање и помагање на машинското учење на автопилотните возила каде што наместо на физички терен симулацијата може да биде изведена во виртуелен свет без несакани последици.

3. Godot – Софрверска платформа за развој на игри со отворен код

Godot (слика 3.1) ги има сите карактеристики опишани погоре што карактеризираат една модерна платформа за развој на игри. Платформата исто така е бесплатна и со отворен код, издадена со MIT [19] лиценца. Ова би значело дека нема цена, скриени трошоци или процентуален трошок. На кратко кажано, играта развиена во Godot би била сопственост на тој што ја развил, што не е случај со многу од други, комерцијални платформи што обврзуваат на различен начин. За многу развивачи на видео игри ова е голема предност.

Исто така, под дефиницијата на софтвер со отворен код се подразбира дека платформата е константно развивана од заедницата која го опколува. Godot постојано се подобрува од заедница на развивачи на софтвер кои имаат голема пасија. Тие го донираат своето време и експертиза со цел понатамошно развивање, тестирање и поправање на грешки. Исто така, заедницата ја гради официјалната документација на платформата.

За развивачите на игри, предностите на Godot се многубројни. Главната предност што еден софтвер со отворен код ја има е достапноста на линиите код од што е составен. Слободата што ви ја дава платформата може да се искористи за цели на играта или пак компанијата. Можност да се надоврземе или да го промениме кодот на платформата и да создадеме дополнителни алатки кои се прилагодни за нашата игра.

Слика 3.1 - Лого на Godot

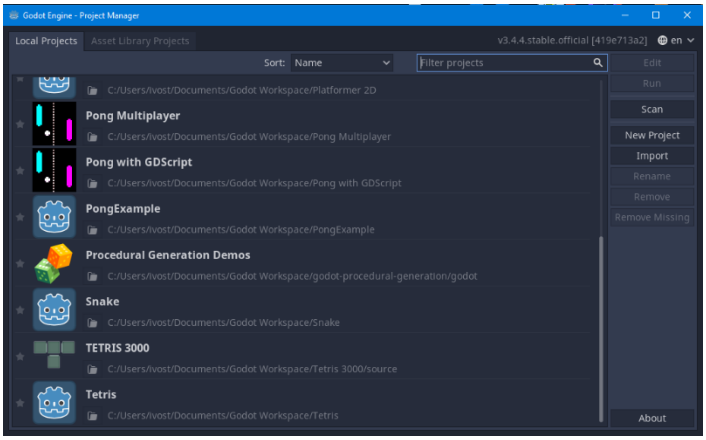
Природата на софтвер со отворен код ни нуди ниво на транспарентност што ја нема кај комерцијалниот софтвер. На пример, ако најдеме некоја грешка во алатките кои ги нуди платформата ние имаме слобода самите да ја поправиме без да чекаме одредена компанија да направи соодветна надоградба. Целосната достапност на кодот на платформата овозможува полесно да дебагираме поголеми проекти.

3.1 Преглед на корисничкиот интерфејс на Godot

Исто како повеќето платформи, Godot користи унифицирана околина за развивање. Тоа значи дека се користи истиот интерфејс за изработка на различни типови на игри или пак на различни аспекти од играта што се развива како: скриптирање, графика, аудио и така натаму. Во понатамошните потпоглавја е даден преглед на интерфејсот на платформата и алатките кои се користат за изработка на играта со одбрана на територии.

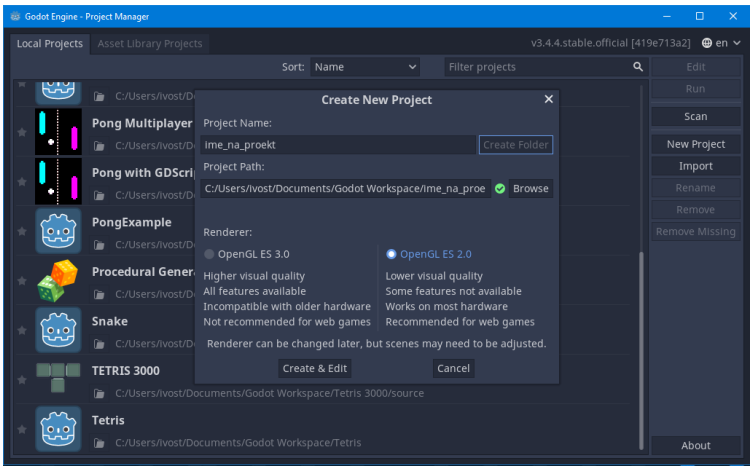
3.1.1 Менаџер на проекти

Менаџерот на проекти (слика 3.2) е првиот прозор што се гледа при стартување на Godot.



Слика 3.2 – Менаџер на проекти на Godot

Во овој прозорец може да се забележат листа од проекти на кои сте работеле во Godot. Има опција да се избере постоечки проект и да се кликне на копчето **Run** со цел да се прикаже финалниот производ од проектот, додека пак кликање на копчето **Edit** пренасочува кон уредувачот.



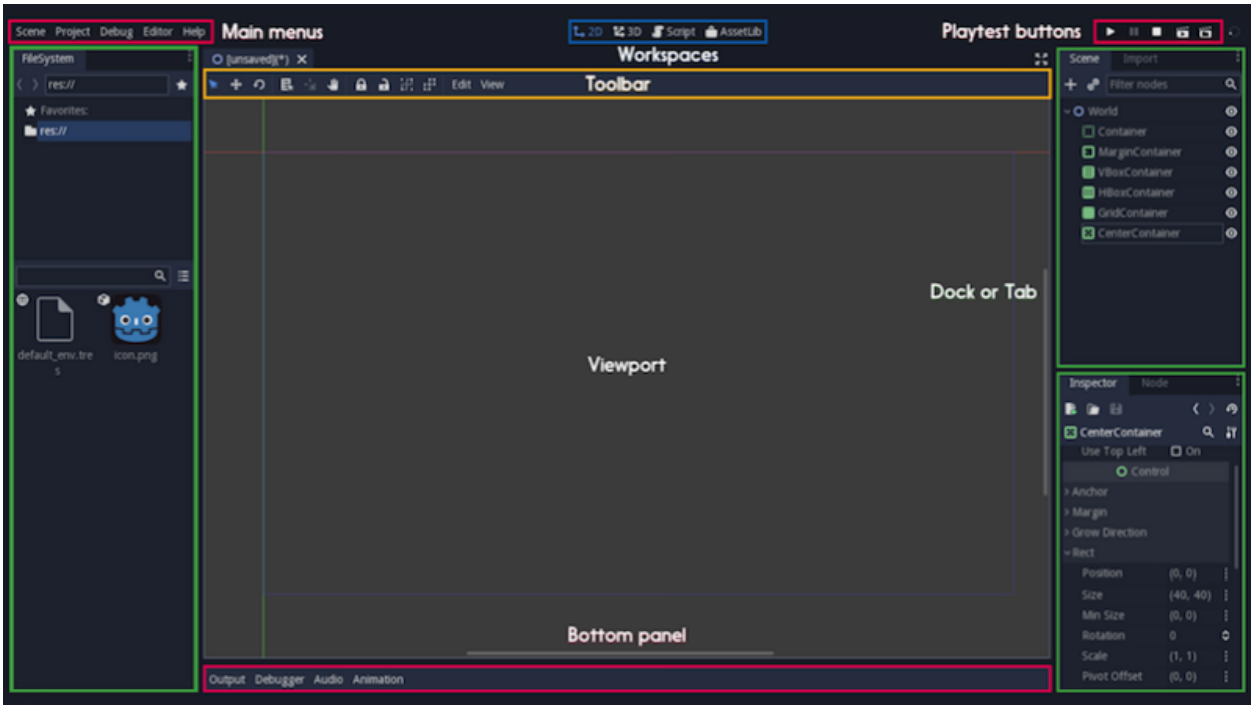
Слика 3.3 - Креирање на проект

Исто така се нуди опција за креирање нов проект со кликање на копчето **New Project**. Модалниот прозорец што се отвора е прикажан на слика 3.3.

Во модалот се внесува името на проектот и локацијата каде треба да се зачува во соодветните полиња. Дополнително, има опција за избор која графичка библиотека да се користи. Godot препорачува OpenGL ES 2.0 за подобра компатибилност со хардвер, додека пак Open GL ES 3.0 нуди повисок визуелен квалитет. Сите потребни датотеки и ресурси кои се користат за развивање на еден проект мора да се дел од датотеката која е наведена. Ова допринесува споделувањето на Godot проекти или подесување на софтвер за контрола на верзии да биде многу едноставно.

3.1.2 Уредувач

На слика 3.4 е прикажан главниот прозорец на уредувачот. Овој прозорец е значаен поради тоа што ќе биде присутен поголемиот дел од времето при развивање. Корисничкиот интерфејс на уредувачот е поделен во повеќе секции притоа секоја секција нуди различна функционалност.



Слика 3.4 - Уредувачот во Godot

Главната секција од уредувачот се нарекува поглед (анг. Viewport), во него се прегледуваат деловите од играта како што се развива. Во горниот лев агол се позиционирани главните менија каде може да се зачува проектот или пак да се вчитаат датотеки, да се направат промени на поставките за проектот и да се побара помош. Во горниот дел со централно порамнување се поставени различни работни површини во кои може да се изработува една игра. Возможно е да се промени погледот помеѓу дводимензионален и тридимензионален свет. Исто така има можност за користење на околина за скриптирање што овозможува промени врз скрипти со вградениот

уредувач за текст. Во опциите на платформата исто така има можност да се вклучи надворешен уредувач на текст како Visual Studio Code.

Табот AssetLib претставува еден вид на продавница во која возможно е да се пребараат многу различни додатоци и готови проекти.

Во горниот десен агол се поставени пет копчиња од кои последователно возможно е да се прегледа играта која се работи, да се паузира извршувањето, да се прекине извршувањето, да се прегледа моменталната сцена или да извршиме сцена по избор.

Во делот обележан со жолто се наоѓа лентата со алатки каде што се поставени најосновните алатки за трансформација. Првите четири алатки овозможуваат единечна селекција, поместување по посакуваната оска, ротација и скалирање. Останатите алатки се еден вид линијар за мерење во пиксели, селекција на сите јазли кои се во групата на еден јазел, прилепување на објекти во однос на ќелии и останати помошни алатки.

Во долниот дел обележан со црвено на сликата 3.4 може да се увидат копчиња кои откриваат прозори за нашата конзола, таб со информации при дебагирање и дополнително алатки за анимација и аудио.

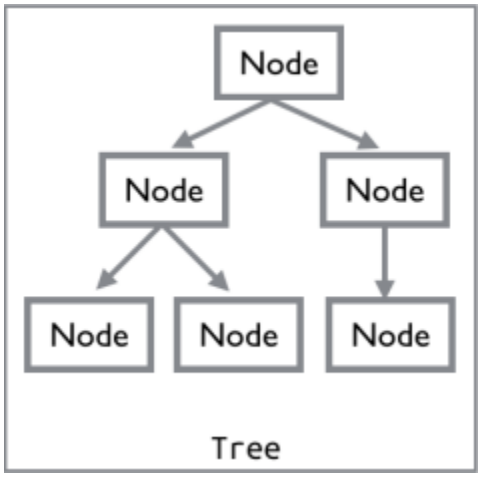
На десната страна обележано со зелено може да се прегледаат јазлите од кои се состои сцената и инспектор. Инспекторот нуди дополнителни информации за селектираниот јазел и променливи кои може да се подесуваат. Опциите се дополнети со опис од документацијата со цел да го едуцираат програмерот за промените што ги прават. При подесување на променлива во инспекторот секогаш има опција да се врати во првобитна состојба.

На левата страна исто така обележано во зелено може да се увиди FileSystem прозорот преку што може да се менаџираат датотеките и ресурсите кои се користат во играта. Прозорот ги прикажува сите датотеки кои се вклучени во проектот и како што напоменав погоре сите датотеки мора да се состојат во главната папка на проектот. Сите ресурси во проектот се наоѓаат релативно на **res://** што претставува коренот на проектот. На пример патека до сцена за главниот играч може да изгледа како **res://scenes/player/Player.tscn** (сцените во Godot ја користат наставката **.tscn**).

3.2 Јазли и сцени

Јазлите се основна градбена единица за создавање на игри во Godot. Јазел е објект што репрезентира најразлични специјализирани функции. Различен тип на јазел може да прикажува слика, анимација или 3Д модел на објект. Јазелот исто така се состои од колекција на карактеристики кои овозможуваат прилагодување на различни поставки. Какви јазли се користат главно зависи од функционалностите кои треба да се постигнат. Тие претставуваат модуларен систем што нуди флексибилност кога се градат делови од игра.

Во дрво структурата новите јазли се додадени како деца на постоечките јазли. Специфичен јазел може да има голем број на деца јазли, но секој јазел мора да има еден родител јазел. Кога се групираат јазли во едно дрво тоа се нарекува сцена и дрвото се нарекува дрво на сцената (слика 3.5).



Слика 3.5 - Дрво од јазли

Godot користи сцени за создавање и организирање на различните објекти во играта. Еден пример за ова е сцена за главниот играч што се состои од јазли и скрипти што се користат за да го направот играчот функционален. Потоа возможно е да постои уште една сцена што дефинира мапа на играта што се состои од препреки кој играчот мора да ги совлада. Потоа се комбинираат дадените сцени во финалниот продукт со користење на инстанцирање за што ќе стане збор понатаму.

Јазли доаѓаат со најразлични карактеристики и функции кои можат да бидат надополнети со додавање на скрипта на јазелот. Скриптата дозволува пишување код што му овозможува на јазелот дополнително однесување. Како пример може да се вметне Sprite јазел во сцена за да се прикаже слика, но ако посакуваното однесување на сликата е да се движи или исчезнува кога ќе биде кликната мора да се додаде скрипта што ќе го овозможи тоа однесување.

Позначајни типови на јазли за 2Д игри се:

- Node2D – примитивен 2D јазел
- Ysort – начинот како слики се преклопуваат во светот
- ParallaxLayer – овозможува паралакс ефект[18]

- Polygon2D – овозможува колизија со многуаголник
- Path2D – јазел што претставува патека
- PathFollow2D – се движи по зададената патека во родител Path2D јазелот
- Light2D – се користи за осветлување во дводимензионален свет

3.3 Програмирање во Godot

Godot користи три официјални јазици за скриптирање: GDScript, VisualScript и C#. GDScript е примарниот јазик вграден во платформата што постојано се развива и претставува препорачаниот јазик за користење. VisualScript е сеуште нов и во фаза на тестирање. Од друга страна C# се цели да се користи во повеќето проекти кога се јавува потреба за подобрување на перформанси. Повеќето проекти во платформата не би имале потреба од подобрено ниво на перформанси, но сепак за тие што имаат потреба Godot нуди флексибилност да користат комбинација од GDScript и C# каде што е потребно.

Освен трите програмски јазици кои се подржани, Godot што е создаден користејќи C++ нуди можност истиот да се користи за развивање со цел да се добие поголема контрола врз перформансите и надополнување на функционалностите на платформата.

Во играта развиена како дел од дипломската работа ќе го користам програмскиот јазик GDScript поради тоа што тој е наменет за користење со Godot и има соодветна поддршка за истиот.

3.4 GDScript

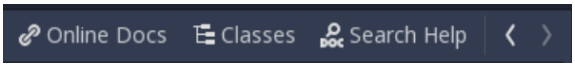
GDScript синтаксата е моделирана врз основа на програмскиот јазик на високо ниво Python. Се смета дека ако веќе го познавате Python многу лесно би можеле да се навикнете на специјализираниот јазик што се користи во платформата. GDScript претставува динамички програмски јазик како Javascript и се смета како лесен за разбирање и учење.

Динамички програмски јазик означува дека немаме потреба да го декларираме типот на променливите кои ги создаваме. Исто така користи празен простор за да го означи телото на функциите. Воглавно користењето на GDScript за програмирање на логиката на игра резултира во пишување на помалку код, што допринесува до побрзо развивање и помалку грешки за поправање.

3.5 Документацијата на Godot

3.5.1 Користење на документацијата во уредувачот

Користење на апликацискиот програмски интерфејс на Godot може да изгледа како предизвик поради големиот број на јазли составени од променливи и методи. Олеснителен фактор е што платформата доаѓа со интегрирана документација што може да се разгледува во самиот уредувач (слика 3.6).



Слика 3.6 - Документација во Script табот на Godot

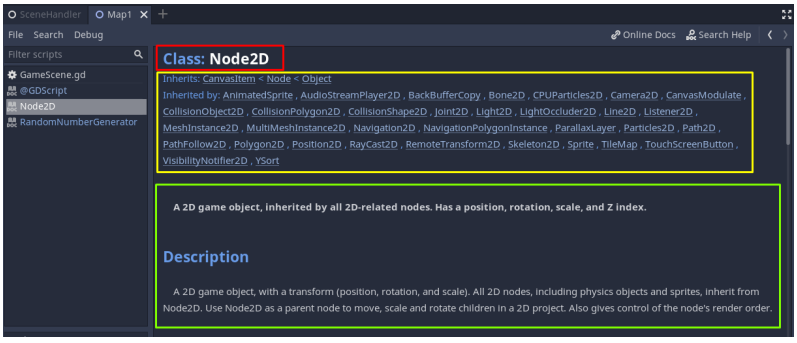
Копчето **Online Docs** ја прикажува документацијата на Godot во веб прелистувач.

Копчето **Classes** овозможува да се разгледуваат достапните јазли и типови на објекти кои се достапни за користење.

Со користење на **Search Help** копчето може да се пребара кое било име на метод или променлива за дополнително објаснување. Двата пребарувачи се паметни што значи може да се внесе само дел од името и резултатите од пребарувањето стануваат се порелевантни на тоа што се пребарува.

3.5.2 Прегледување на документацијата

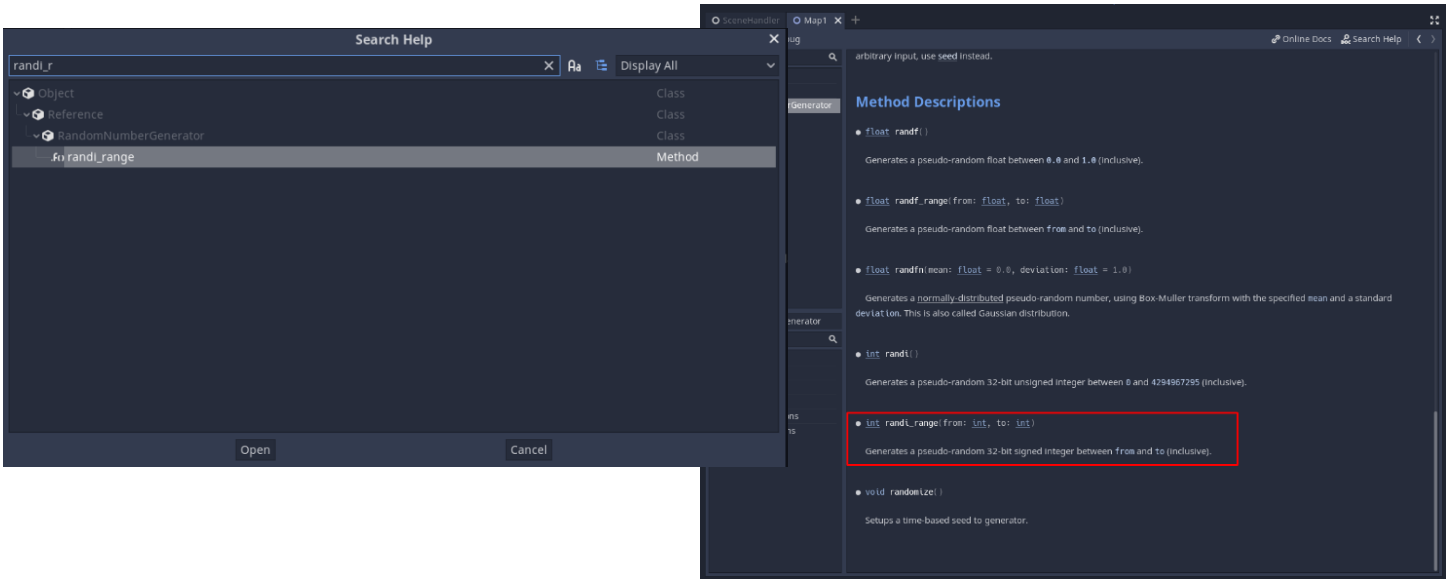
При пронаоѓање на променлива или метода што сте ја барале кликнете на копчето Open и референцата до документацијата ќе се отвори како што е прикажано на сликата подолу (Слика 3.7). Документацијата следи одреден формат што ја прави полесна за читање, односно на врвот го имате името на јазелот/класата и опис за истиот. После името следи секција за променливите кои се користат во таа класа, секција каде се излистани методите, лекции што го објаснуваат јазелот и уште две дополнителни секции каде имаме детален опис на методите и променливите кои се присутни во јазелот.



Слика 3.7 - Со црвено е обележан насловот со името на јазелот, со жолто е обележано од кои класи наследува јазелот и кои класи наследуваат од него, со зелено е обележан описот на јазелот

На слика 3.7 може да се види дека документот започнува со синџир од класи. Синџирите секогаш завршуваат со Object што претставува базичната класа на Godot. Ова овозможува да се види кои карактеристики нашиот тип на објект има. Може да кликнеме на кој било јазел во синџирот за да ги прегледаме неговите детали.

Листата дадена на слика 3.8 прикажува листа од јазли кои наследуваат од моментално прикажаниот јазел. Потоа следи описот на јазелот проследено со секциите за променливите и методите кои ви се достапни. Методите и променливите се подвлечени и означуваат хиперлинк до подолните секции каде може подетално да се разгледаат истите.



Слика 3.8 - Пример за пребарување на метода во интегрираната документација на Godot

Посетувањето на документацијата за апликацискиот програмски интерфејс на Godot е препорачливо со цел добивање попрецизно разбирање како платформата работи.

4. Практична изработка на видео игра во Godot

Со цел да се илустрираат можностите на Godot, во рамки на оваа дипломска работа е изработена игра од видот одбрана на територии (слика 4.1). Овој вид игри се поджанр на стратегиските видео игри и најчесто се изведени со база што припаѓа на играчот и непријатели кои се движат по одредени патеки кои се јасно видливи. Со цел играчот да ја одбрани својата база тој треба покрај патот да гради кули за одбрана.

Напредувањето најчесто се обележува со бројот на бран во кој се наоѓате, односно што поголем бран така правопропорционално се зголемува моќноста на непријателите.

Слика 4.1 - Пример за Tower Defense игра

4.1 Дефинирање на видео играта

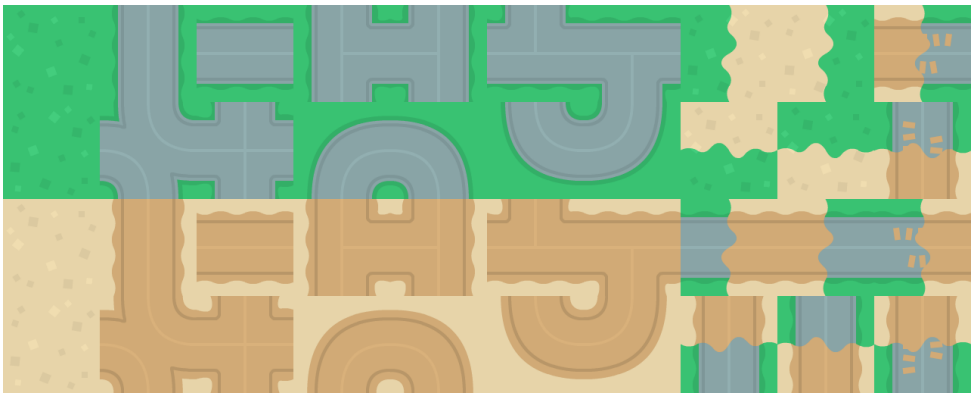
Со цел изработка на прототип на игра со одбрана на територии, како програмер нема да се задржувам на процесот на создавање на графика или музика, односно ќе се фокусираме на процесот на создавање на прототип.

Но пред да се премине на создавањето на прототипот треба првично да се создаде план и некоја првична дефиниција за играта.

Со цел играта да припадне на овој жанр потребни се непријатели, патека по кои ќе се движат, кули кои ќе ги спречат непријателите и на крај базата на играчот.

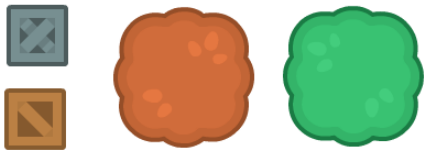
Слика 4.2 - Тенк

Непријателот е претставен како син тенк (Слика 4.2). Како што е познато тенковите најчесто се движат по терен или пак по асфалт. Во играта со цел да биде јасно видлив нивниот пат на движење оптимално би било тенкот да се ограничи со движење само на асфалт.

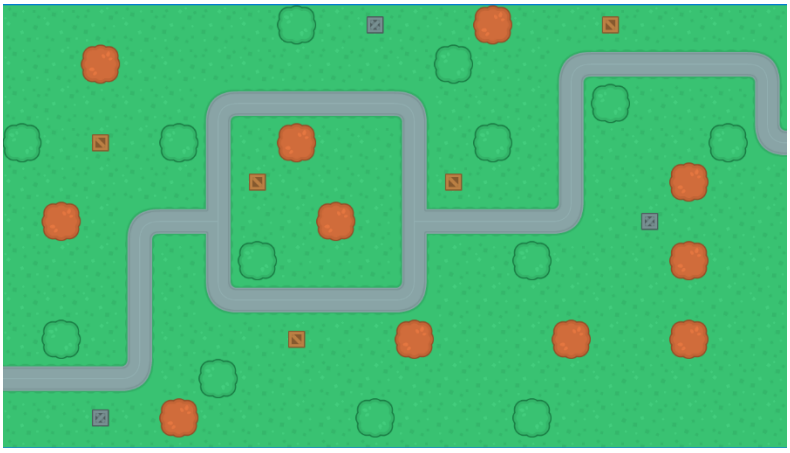


Слика 4.3 - Слики од пат и терен за избојување на мапа

Со користење на слика (анг. Tilesheet), може да се создаде цела мапа со чист терен и пат (слика 4.3). Но, ако мапата се состои само од терен наспроти пат тогаш таа би била монотона. Со цел ова да се смени, може да се додадат и декорации (слика 4.4).



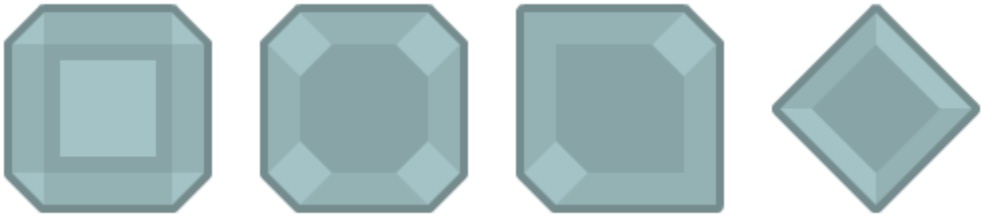
Слика 4.4 - Декорации



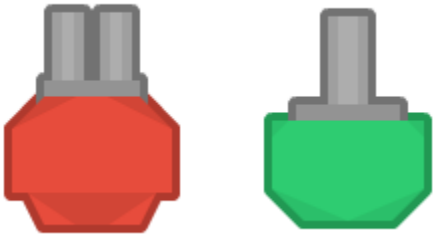
Слика 4.5 - Целосна мапа

При комбинирање на сликите заедно со декорациите, се добива една целосна мапа (слика 4.5). На ваков начин може да се изградат бесконечен број мапи што би ја направило играта подинамична од аспект на тоа што секој пат би требало играчот да се справи со самата поставеност на мапата.

Интеракцијата на играчот би се состоела од тактички градење на кули со цел спречување на непријателот да ја уништи базата. Таа не е физички прикажана, туку ако левиот дел од мапата се смета за влез на непријателите десниот дел би означувало судар со базата. Штетата што би ја правеле тенковите ќе се прикажува преку корисничкиот интерфејс на играта, но за тоа ќе зборуваме подоцна.

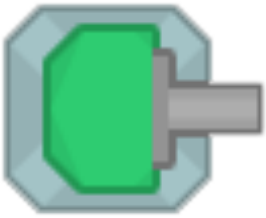


Слика 4.6 - Лежиштето на кулата



Слика 4.7 - Глави на кулите

Кулите за одбрана во играта се претставени со лежиште (слика 4.6) и глава на кулата (слика 4.7). На ваков начин, одвоените слики даваат поголема слобода за правење комбинации. Лежиштето може да означува ниво на кулата, додека пак главите да ни означуваат тип на кула што ја користиме.



Слика 4.8 - Изглед на основната кула

Наведените детали даваат чиста претстава за тоа како би изгледала играта, како би се одвивала интеракцијата од страна на играчот и кои се основните механики што треба да се изградат.

Тенковите треба да се движат по одредени патеки, патеките се јасно видливи, но за да се направи играта поинтересна може да се додаде можност на тенковите по случаен избор да ја изберат својата

патека односно ако се разгледа сликата 4.5 можеме да се заклучи дека тенковите можат да се движат по горниот и долниот пат, но влезот во мапата сепак останува од левата страна. Ова му дава можност на играчот да постави одбрани по двете патеки со цел да не пропушти тенкови да навлезат во неговата база.

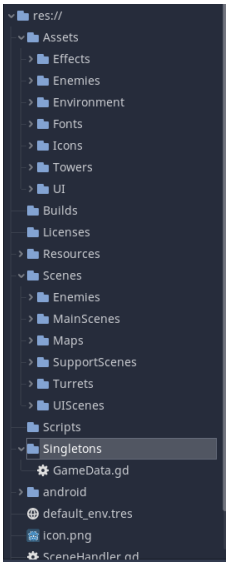
Тенковите имаат живот на кој полека кулите на играчот им прават штета. Со цел создавање на предизвик на играчот тенковите секој бран ќе добиваат поголема брзина на движење и поголем живот. Балансот во играта ќе се постигне со тоа што на играчот му овозможува да стане посилен. По секое уништување на тенк следува награда што во играта ќе биде претставена со пари. Со тие пари играчот одлучува дали ќе ги потроши на надоградување на веќе постоечките кули или пак ќе бидат изградени нови.

Начинот како играта ќе се истакнува е со воведување на строго дефиниран random number generator систем што ќе ја направи играта динамична и непредвидлива. Односно брзината и животот на тенковите ќе биде врзан со броеви по случаен избор чиј домен постепено ќе се зголемува соодветно на нивото во кое се наоѓа играчот.

4.2 Поставки на проектот

4.2.1 Хиерархија на ресурсите во проектот

Веднаш по креирањето на проектот многу е важно соодветно да се организира (слика 4.9). Кога се работи на проект во што се користат најразлични датотеки за графика, музика, скрипти, сцени и конфигурациски фајлови да се групираат во директориуми.



Слика 4.9 - Хиерархијата на датотеки во проектот

Директориумите треба да се именувани на дискриптивен начин со цел полесно пребарување на потребната датотека. Пример во случај ако е потребно да се пронајдат одредени скрипти интуитивно би се обратиле кон фолдерот “Scripts”, додека, пак, ако потребна дводимензионална

слика од тенк што го претставува непријателот во играта, би би требало да се бара во “Assets/Enemies”.

Ваквата структура на папките овозможува ефективност и заштедување на време при работење на проект.

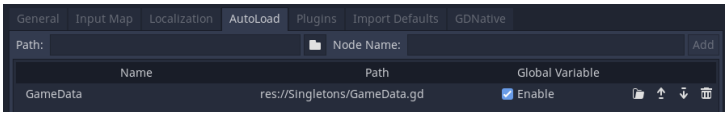
4.2.2 Скрипти што автоматски се вчитуваат

Користењето на скрипти што автоматски се вчитуваат е присутно во повеќето софтверски рамки. Карактеристично за овие скрипти е што може нивните методи и променливи глобално може да се користат. Во играта ова е корисно место да се дефинираат карактеристиките на кулите.

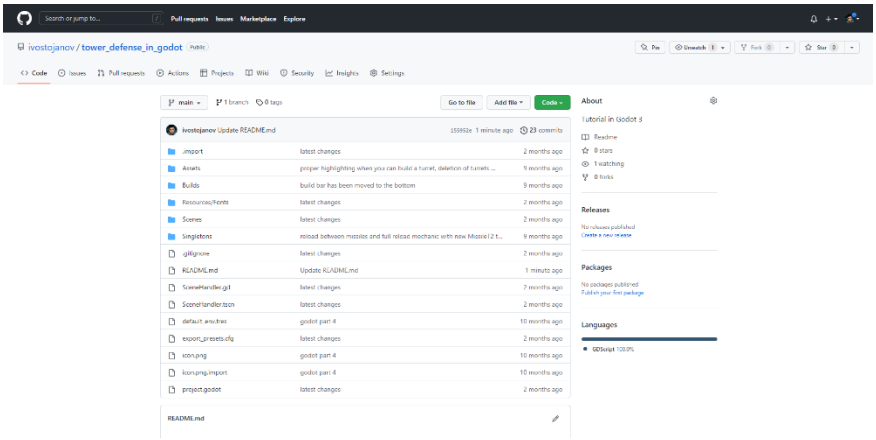
Начинот како се создава и врзува ваков вид скрипта е исто како и создавање на нормална скрипта што наследува од надкласата Node, потоа во таа скрипта во форма на објект се внесени деталите во случајов карактеристиките кои ги поседуваат нашите кули (слика 4.10).

Слика 4.10 - Карактеристиките на кулите во GameData.gd

Со цел ново креираната скрипта да се вчита на почетокот на самото извршување и да биде глобално достапно потребно е да се додаде скриптата во autoloading делот на поставките на проектот (Слика 4.11).

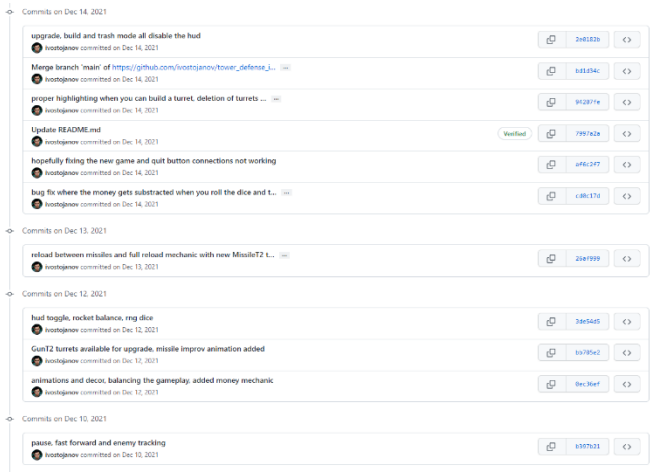


Слика 4.11 - Додавање на скриптата GameData.gd во секцијата autoloading



Слика 4.12 - Tower Defense In Godot репозиториумот хостиран на Github

Поставување на софтвер за контрола на верзии е корисна алатка што најчесто се користи при развивање на софтвер од страна на тим (слика 4.12). Повеќе програмери во исто време можат да придонесуваат на еден централизиран репозиториум. При изработка на играта со одбрана на кули софтвер на контрола за верзии е од помош да може да се вратам назад на претходни верзии со цел спречувањето на грешки што можат да се јават при развивање на играта. Репозиториумот за играта е хостиран на Github (можете да се пристапи на овој [линк)](#).



Слика 4.13 - Пораките оставени за секој commit

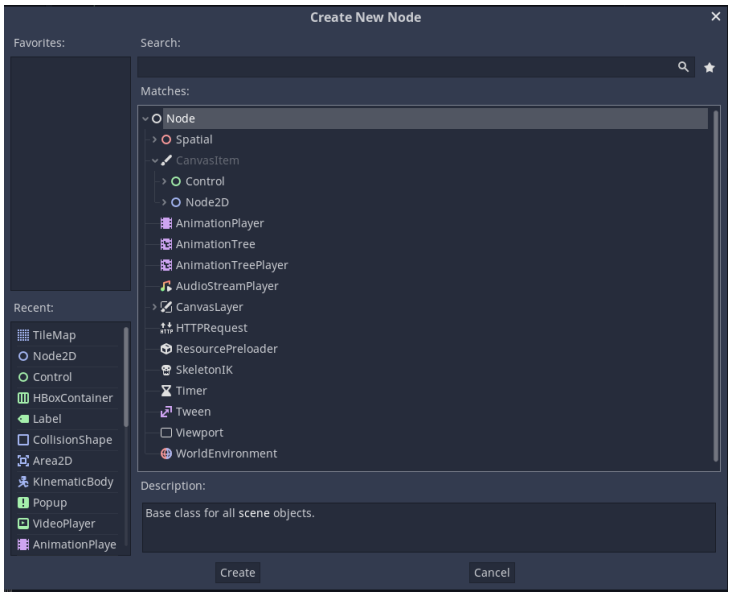
Пораките кои што се оставени покрај секој комит се дискриптивни на модулот што се развива или се однесуваат на решавање на грешките што се јавиле при развивање (слика 4.13).

4.3 Создавање на прототип

Во следните поглавја ќе бидат прикажани алатките и јазлите кои се користат за да се создаде видео играта и нејзините клучни карактеристики како што се менаџирање на сцените, развивање на кулите за одбрана, создавање на систем за изградба и однесувањето на непријателот. Пропратно ќе биде разгледано и како броевите по случаен избор ја прават играта динамична и интересна за играње.

4.3.1 Менаџирање на сцените

Структурата на јазлите во еден проект е индивидуална, но мора да има смисла и да го олесни начинот како се менаџираат јазлите кои ќе бидат вчитани и отстранети. Под коренот на играта е додадена сцена што претставува јазел (Node) што се вика “SceneHandler” (слика 4.14).



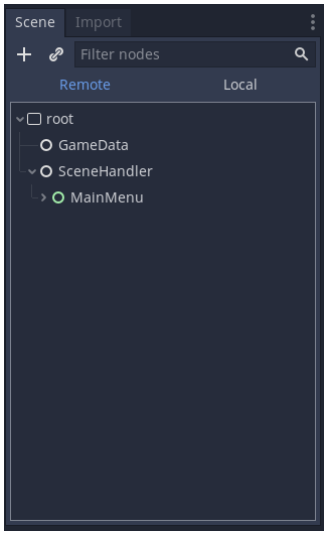
Слика 4.14 - Создавање на јазел (“Node”)

На истиот јазел е додадена и скрипта со истото име. Преку оваа скрипта се извршува менаџирањето на сцени. Моментално играта има две сцени едната е менито од што се започнува играта и друга сцена што претставува основата на играта.

SceneHandler јазелот поради неговата улога да ги менаџира јазлите ќе биде постојано активен. При негово создавање тој прави инстанцирање на јазел што е одговорен за главното мени. Комуникацијата помеѓу јазлите MainMenu и SceneHandler се одвива преку сигнали. Во Godot можеме ние сами да создадеме сигнали или пак да користиме предефинирани сигнали кои ги имаат јазлите.

При инстанцирање на јазелот што е одговорен за целата логика на играта јасно ни станува дека веќе не е потребно главното мени поради тоа што ја има завршено својата задача. Поради таа причина со функцијата queue_free() јазелот се додава во редица за бришење.

Таа редица осигурува дека јазелот со сите негови деца јазли е безбедно избришан до крајот на извршувањето на моменталната рамка со што ослободуваме меморија.



Слика 4.15 - Дебагирање на хиерархијата на јазли при извршување

GameScene ги наследува карактеристиките на Node2D јазелот и за него е прикачена скрипта наречена GameScene.gd што е одговорна за целата логика на играта за што ќе зборуваме подоцна.

4.3.2 Развивањето кули за одбрана

Кулите за одбрана кои се користат во играта се поделени во две категории. Едната се состои од обични кули кои би правеле штета и потоа би имале период во секунди каде што ќе бидат оставени да ја наполнат својата муниција. Оваа кула ќе ја наречеме “Gun” кула што ќе има една надоградба што ќе ја зголеми штетата што ја прави и далечината на што може да стрела.

Можеме да ги разгледаме карактеристиките подетално во следново набројување:

- damage – означува штетата што ја прави кулата
- rof – претставува колку често кулата испукува во една секунда (анг. Rate of Fire)
- range – далечината на што кулите можат да ги видат своите непријатели, во пресметката ова означува радиус на кружницата што претставува далечина измерена во пиксели
- category – означува категорија на оружјето што го користи во случајов имаме ракети или проектили
- cost – претставува цената за изградба на дадената кула
- upgradable – претставува променлива што дава значење на нашиот систем за надоградби дали таа кула е возможно да се надогради

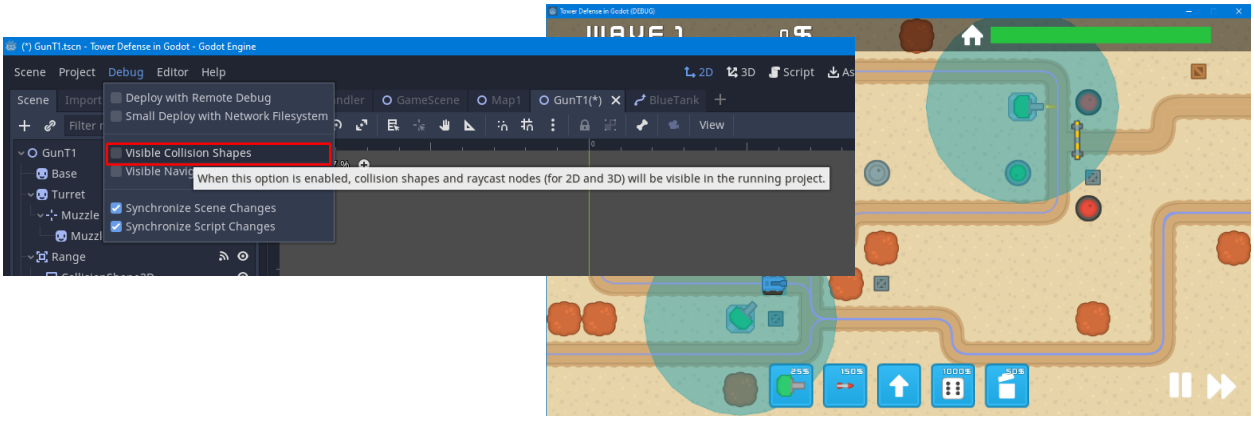
Кулите со ракети споредбено со основните “gun” кули ќе користат интересна механика инспирирана од играта World of Tanks што претставува симулација на војна со тенкови кои имаат реалистични карактеристики за движење и стрелање. Механиката наречена “autoloader” кај еден тенк овозможува да испукаме повеќе проектили со минимална пауза меѓу испукувањето, а притоа

да бидеме казнети за истото со подолго време за полнење на муницијата. Во контекст на оваа механика нашата кула со ракети ќе има можност да испука 3 ракети со минимална пауза од 500 милисекунди помеѓу испукување но притоа ќе биде казнета со долго чекање од 3 секунди за испунување на сите ракети.

Со цел да се изведат кулите е создадена базична скрипта наречена Turrets.gd од што се наследува. Во оваа скрипта се создадени основните а и воедно взаемните карактеристики кои ги поседуваат кулите. Добар пример за ова е начинот како се бира непријателот што ќе му биде мета за уништување. Непријателот се одредува со помош на offset вредноста што одредува колку од зададената патека ја има поминато во пиксели.

Друга функција што ќе ја користат нашите кули е вртење, односно ротација кон својот непријател.

Пред да го избереме непријателот треба да најдеме начин како да ги избереме непријателите кои се во радиусот на видик на кулата. Првично додаваме Area2D јазел што ја одредува физиката на објектот и дете јазел CollisionShape2D што го одредува обликот на објектот, во случајот користиме облик од тип CircleShape2D што ќе го означи радиусот на нашите кули. Кога тенк ќе влезе во површината што ја обележува радиусот се активираат сигналите body_entered и body_exited.



Слика 4.16 - Дебагирање на јазлите кои се користат за физика во Godot

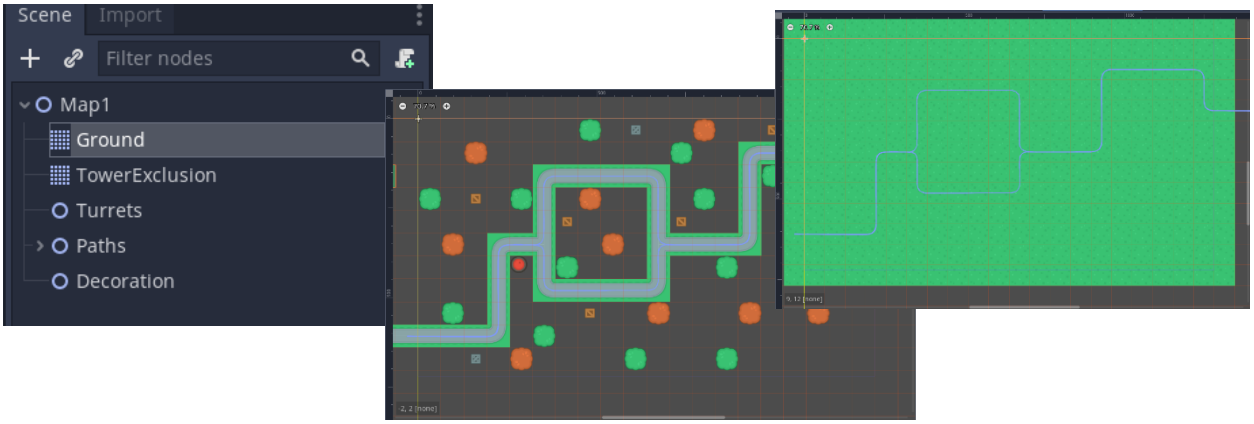
На слика 4.16 е прикажан начин како може при извршување на сцена/игра да се гледаат и дебагираат јазлите кои ги користиме за физика. Во случајот може да се видат патеките по кои се движат тенковите и радиусите на кулите до каде можат да стрелаат и соодветно дали пресметките кои ги правиме доведуваат до очекувани резултати.

Останатите функции за стрелање и за обновување на муниција ќе бидат предефинирани во класите кои ќе се користат за различните кули. Причината за предефинирање на овие методи е тоа што стрелањето, механиката за обновување на муниција и анимациите ќе бидат уникатни за секоја кула.

4.3.3 Создавање на систем за изградба, уништување и надоградување на кули

Следниот систем што се разгледува е клучен при создавање на игри на одбрана на територии. Изградбата на кули треба да можеме да ја извршиме на терен што е празен. Во овој случај тоа значи теренот да нема препреки, под препреки подразбираме пат, кутија, дрво или пак постоечка кула. Ваквото однесување ќе се постигне со користење на Tilemap јазел.

Мапите што се создадени се состојат од два Tilemap јазли, едниот е обележан со земја, трева или пак песок што ќе претставува празен простор (слика 4.17). Додека пак вториот Tilemap јазел го обележува зафатениот дел каде што не може да се гради, истиот се користи во скриптата за да се одреди дали е возможно во посакуваниот квадрат да се гради.



Слика 4.17 - Од лева страна е дадена хиерархијата на првата мапа, во средина изгледот на мапата TowerExclusion, додека, пак, десно е дадена мапата Ground

Градењето на кули се ограничува на кули од прво ниво со цел да се мотивираат играчите да експериментираат со сопственото градење. Иницијализирањето на процесот на градење започнува со тоа што корисникот ќе кликне на една од двете можни кули во долното мени на корисничкиот интерфејс (слика 4.18).

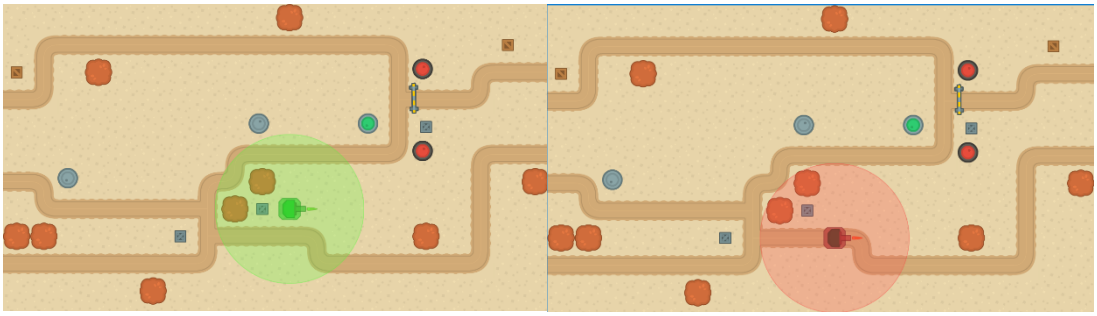
Додека пак останатиот дел од head-up display-от е од информативен карактер, односно во горниот дел има црно транспарентна лента на што го пишува нивото во кое се наоѓа играчот, парите што тој ги собрал за градење и животот на нашата база што е прикажан со прогрес бар. Прогрес барот со што е прикажан животот може да се истакне поради тоа што користи TextureProgress јазел на што е прикачен дете јазел наречен Tween, што овозможува едноставно анимирање на зададена

променлива. Односно наместо нагло да се промени вредноста таа линеарно се намалува до посакуваната вредност создавајќи еден тип на анимација.

Слика 4.18 - Корисничкиот интерфејс на играта, со црвено е обележано менито за градење

Менито за градење се состои од следниве копчиња:

- GunT1 копче – иницијализира градењето на обична кула од прво ниво
- MissileT1 копче – иницијализира градење на кула со ракети од прво ниво
- Приказ за надоградба – прикажува мени над секоја кула со информативна стрелка што е бледа или обоена во однос на тоа дали таа кула може да се надогради
- Коцка – играње со генератор на броеви по случаен избор, активира анимација на коцката и во случај ако се добие бројот шест, животот на базата се обновува
- Канта – иницијализирање на мени каде што може да се уништат кули



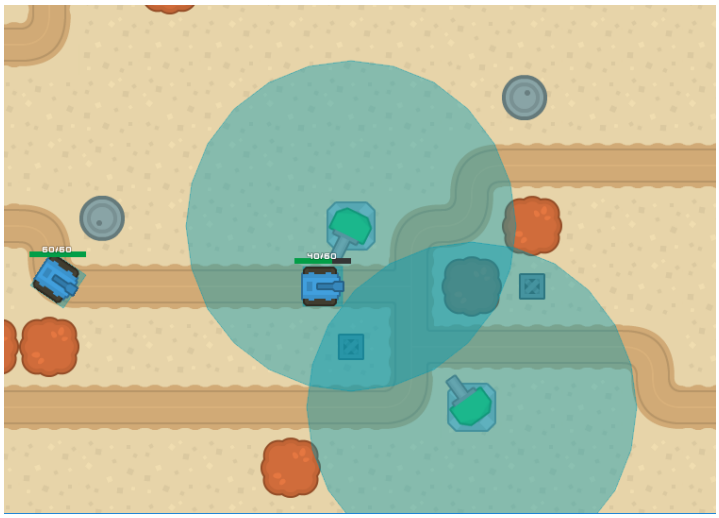
Слика 4.19 - Изградба на кула на слободен терен (можно) наспроти изградба на кула на пат (забрането)

Главниот предизвик што се јавува при развивање на системот за изградба е тоа што самиот кориснички интерфејс оневозможуваше да се изградат кули по желба. Поради таа причина одлучив да го кријам целиот head-up display додека е во тек изградбата на кулите. Друг, поголем, предизвик што се појави беше начинот на одредување на празниот простор на кој може играчот да гради, чие решение се состои од боење на транспарентен sprite во полиња каде што има изградено кула со цел јазелот Tilemap да врати вредност различна од -1, што означува дека полето е окупирано од објект (слика 4.19).

4.3.4 Непријателот и неговото однесување

Непријателите во играта се претставуваат како тенк што се движи по предодредена патека. Во различните мапи има различен број на патеки. При нивното инстанцирање тие по случаен избор се движат по одредена патека.

За тенковите има прикачено јазли KinematicBody2D со дефинирано тело за колизија кое е CollisionShape2D со облик на правоаголник. Колизијата на тенковите е единствено искористена да се детектира кога тие навлегле во радиус на кулите на играчот (слика 4.20).



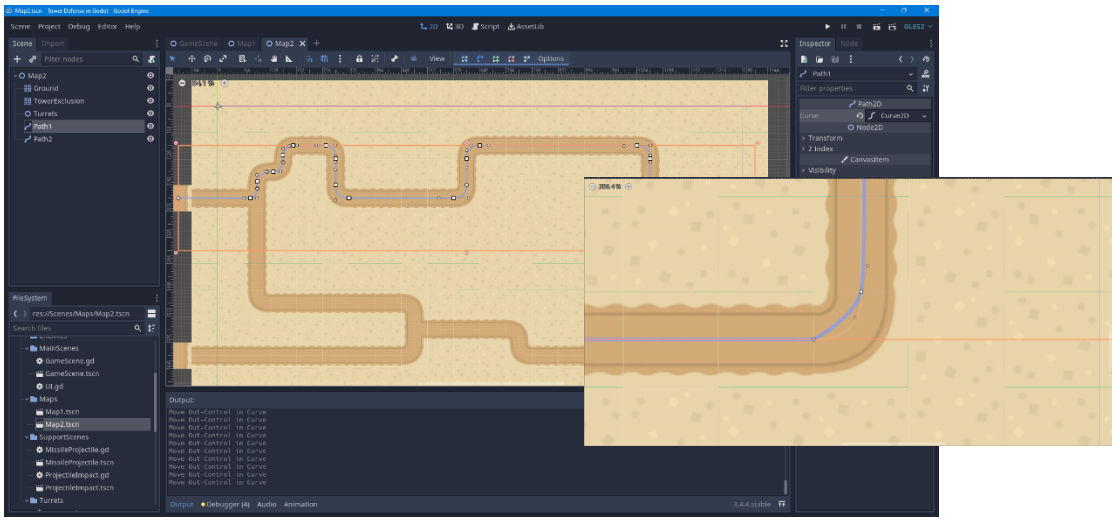
Слика 4.20 - Правоаголникот на колизија на тенкот и детекцијата од страна на кулите

Главен (родителски) јазел на еден тенк претставува јазелот PathFollow2D што е потребно да се инстанцира во PathNode2D со цел да започне неговото следење со некаков offset зададен во пиксели. Детектирањето на колизијата на тенкот со нашата база се прави со помош на параметарот unit_offset на PathNode2D јазелот што означува вредност од 0 до 1 соодветно почеток и крај на патеката.

Со цел играчот да добие некој тип на повратна информација кои од тенковите се нападнати додадена е кратка анимација што се состои од 5 слики што се појавуваат на позиција по случаен избор на површината на тенкот (Слика 4.21).

Слика 4.21 - Анимација на експлозија кога се одзима живот на тенкот

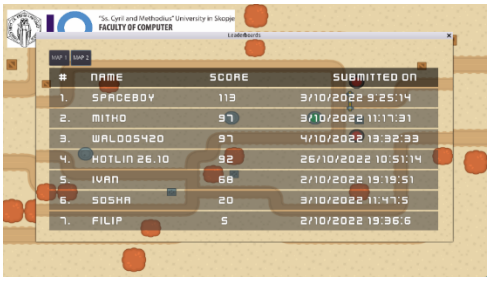
Погоре напоменав дека тенковите се од тип PathFollow2D јазел и дека им е потребен татко јазел што е од тип Path2D. Овој тип на јазли се претставени со криви кои можеме рачно да ги модифицираме. На сликата подолу (слика 4.22) може да се видат клучните точки и како еден пат е исцртан.



Слика 4.22 - Исцртување на патека по што се движат тенковите

4.3.5 Табела за натпреварување

Со цел играчите во играта да имаат поголем мотив да ги совладуваат нивоата, се имплементира систем за натпреварување. Овој систем користи позадински сервис кој зачувува поени, време кога се направиле поените и име на чие припаѓаат. Резултатите во табелата нема да се асоцираат со кориснички профили поради тоа што немаме автентикација, односно ќе бидат асоцирани со внесено име. За секое име се зачувува најголемата вредност на поени, останатите се бришат.



Слика 4.23 - Корисничкиот интерфејс на табелата за натпреварување

Корисничкиот интерфејс за табелите за натпреварување (слика 4.23) се состои од две копчиња кои овозможуваат вчитување на моменталните информации внесени на Silent Wolf сервисот за двете создадени мапи. Резултатите се подредени по опаѓачки редослед на поени пропратени со името на играчот неговата позиција на табелата и времето кога ги направил поените.

4.3.6 Генерирање броеви по случаен избор и динамичноста на играта

Начинот како оваа игра се истакнува од останатите направени во Godot или пак во истиот жанр е со стриктно користење на генератор на броеви. Користење на броеви по случаен избор ја прави играта да биде непредвидлива односно брзината на тенковите, нивниот живот, штетата што ја прават кулите или пак заработувачката се зависни од броевите по случаен избор.

Се разбира овие броеви по случаен избор се во одреден домен со цел самите карактеристики на кулите и тенковите да не изгубат смисла, но и воедно да им дадат уникатност. Многу е мала веројатноста да најдете две кули или два тенка што би биле од ист тип да имаат исти карактеристики.

Користењето на ваква динамичноста се одразува на **позитивен** и **негативен** начин. Односно **позитивниот** претставува тоа што играта никогаш не е иста за играње, некогаш играчот ќе има шанса да изгради една кула до петото ниво но кога можеби има повеќе среќа градењето би се одвивало и одма по второто ниво.

Негативниот аспект се состои во тоа што со цел ваквата игра да има еден вид баланс потребно е прекумерно тестирање. Тестирањето во случајов беше направено од неколкумина волонтери кои ги потенцираа потешкотиите кои ги имаат при играњето. Односно некогаш играта не била “фер” спрема нив додека пак друг пат била полесна од вообичаено за играње. Моментално балансот е доведен до прифатливо ниво иако сепак има недостатоци.

5. Заклучок

Со дипломска работа беа илустрирани дел од можностите на софтверската платформа за развој на игри Godot со изработка на игра од жанрот на одбрана со кули.

Во понатамошна надградба на играта може да се користат броеви по случаен избор кај кулите со цел да се направи подоцнежниот период од играта по интересен. Останати промени може да претрпи кантичката на која може да се додаде карактеристиката да руши терен на кој не може да се гради (пример: дрва, буриња или кутии) со цел на играчот да му се овозможи поставување нови кули. Имплементација на автентикација со цел играчите да ги поседуваат своите резултати и исто така може да се замени позадинскиот сервис Silent Wolf со сопствено решение.

6. Референци

[1] Jason Gregory. Game Engine Architecture. A K Peters/CRC Press, 2019. isbn: 9781138035454.

[2] Godot Game Engine, url: https://godotengine.org/

[3] Raph Koster. A Theory of Fun for Game Design. Phoenix, AZ: Paraglyph, 2004.

[4] Johan Huizinga. Homo Ludens. Random House, 1938. ISBN: 9780807046814.

[5] id Software LLC, url: https://en.wikipedia.org/wiki/Id_Software

[6] High-level programming language, url: https://en.wikipedia.org/wiki/High-level_programming_language

[7] RPG Maker, url: https://www.rpgmakerweb.com/

[8] List of games made in the Unreal Game Engine, url:

https://en.wikipedia.org/wiki/List_of_Unreal_Engine_games

[9] Serious Engine, url: http://www.croteam.com/technology/

[10] Transform, clipping and lighting, url: https://en.wikipedia.org/wiki/Transform,_clipping,_and_lighting

[11] Crash Team Racing, url: https://crashbandicoot.fandom.com/wiki/Crash_Team_Racing

[12] Fighting Games Genre, url: http://en.wikipedia.org/wiki/Fighting_game

[13] Northgard, url: https://northgard.net/

[14] Heaps.io, url: https://heaps.io/

[15] Виртуелна продукција на првата сезона на Mandalorian – url: https://youtu.be/gUnxzVOs3rk

[16] Reimagine the automotive lifecycle with real-time 3D | Unity – url: https://youtu.be/mj7TYBisWB8

[17] Definitions and Applications of Augmented/Virtual Reality, url: https://www.researchgate.net/publication/350108592_Definitions_and_Applications_of_AugmentedVirtual_Reality_A_Survey

[18] Parallax Scrolling in Godot 3.2, url: https://www.youtube.com/watch?v=f8z4x6R7OSM

[19] Open Source Initiative – The MIT License, url: https://opensource.org/licenses/MIT

[20] Chris Bradfield. Godot Game Engine Development Projects. Packt, 2018 ISBN:

[21] Kenney ресурси, url: https://www.kenney.nl/

[22] Game Development Center, url: https://www.youtube.com/c/GameDevelopmentCenter

[23] Godot Documentation, url: https://docs.godotengine.org/en/stable/

[24] Modding, url: https://en.wikipedia.org/wiki/Modding