

ПЛОВДИВСКИ УНИВЕРСИТЕТ
“ПАИСИЙ ХИЛЕНДАРСКИ”

Факултет по Математика и Информатика
Катедра “Компютърни системи”
Специалност “Софтуерно инженерство”

Дипломна Работа

на тема:

“XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX”

Дипломант:
XXXXXXXXXXXXXXXXXXXX
Ф. номер XXXXXXXXXX

Научен ръководител:
XXXXXXXXXXXXXXXXXXXX

Пловдив 2023

Съдържание

Увод.....	4
Цел и задачи.....	6
Използвани технологии.....	7
Unreal Engine.....	7
Midjourney.....	9
GIMP.....	11
Sony Vegas Pro 15.....	12
Архитектура.....	13
Тип.....	13
Размер.....	13
Основна Функционалност.....	13
Система за движение на главния герой.....	13
Система за поемане и възстановяване на щети.....	15
Система за динамична камера.....	17
Система за избягване на определени атаки.....	18
Система за хвърляне на обекти.....	19
Система за диалог.....	20
Система за запазване на прогрес.....	21
Система за запазване на рекордно време.....	21
Архитектура на менютата.....	22
Обекти и взаимодействия между тях.....	23
Разработка на играта.....	25
Процес на създаване на графиката.....	25
Разработка на системите на играта.....	29
Система за движение на главния герой.....	29
Система за поемане и възстановяване на щети.....	33
Система за динамична камера.....	34
Система за избягване на атаки.....	35
Система за хвърляне на обекти.....	36
Система за диалог.....	37
Система за запазване на прогрес.....	40
Система за запазване на рекордно време.....	42
Имплементация на менютата.....	43
Непреценени разработки.....	44
Неоточнени разработки.....	44
Тръни.....	44

Диво прасе.....	44
Катерици.....	45
Летящи тръни.....	46
Змия.....	46
Плодов храст.....	47
Базов терен.....	47
Невидима стена.....	48
Летяща платформа.....	48
Движеща се платформа.....	49
Непредвидени разработки.....	49
Обучаващи невидими колизии.....	49
Други.....	49
Последната битка.....	49
Анимации.....	50
Ръководство на игрacha.....	52
Начало на играта.....	52
Първо ниво.....	53
Второ ниво.....	54
Трето ниво.....	54
Четвърто ниво.....	56
Пето ниво.....	57
Шесто ниво.....	58
Седмо ниво.....	59
Осмо ниво.....	60
Заключение.....	61
Бъдеще на играта.....	61
Библиография.....	62

УВОД

Сферата на компютрите и компютърните науки е една от най-бързо разрастващите се в съвременния свят. Наблюдава се бърз темп на развитие, промени, конкуренция, еволюция и иновации, както при хардуер, така и при софтуер. Винаги има някоя нова идея или разработка, която е на път да разтърси това, което си мислим, че знаем или начините, по които компютрите се използват днес. Постоянно се разширяват граници и се намират решения на проблеми, които преди това са смятани за нерешими.

Изминали са по-малко от 50 години от известното изказване на Кен Олсън, че никой няма нужда от компютър в дома си и то вече може да се смята за едно от най-грешните предсказания в историята на човечеството. Компютрите във всичките си форми вече са неделима част от съвременния начин на живот. Сферата на компютрите се пресича с много други сфери, които сформират основата на нашата цивилизация. За наше щастие това означава, че всяка от тези други сфери може да се възползва от бързият темп на подобрене и иновации, за да подпомогне своето развитие. Тази помощ, която компютрите оказват е толкова голяма, че при някой сфери тя се равнява на векове прогрес постигнат само за няколко години.

В историята на компютрите можем да идентифицираме ключови моменти, които представляват голям скок във възможностите им. Моменти като замяната на вакуумните тръби с далеч по-компактни и продуктивни транзистори, появата на личните компютри за вкъщи, Интернет, първите многоядрени процесори и появата на видео картите представляват толкова голяма промяна във способностите на компютрите, че можем да ги смятаме като собствени ери в тяхната история.

Днес сме на прага на следващата ера в тази кратка история. Но каква е иновацията, която ще промени всичко? В последните години може да се проследи голямо развитие в хардуерните способности на видео картите. Тези нови способности отключиха потенциала на една сфера на компютърните науки, която доскоро беше сравнително малка. Разбира се, става въпрос за невронни мрежи, машинно учене и изкуствен интелект.

Основното препятствие, което стоеше пред развитието на невронните мрежи, беше нуждата от изпълняване на огромно количество подобни изчисления, което изискваше суперкомпютри за по-големите невронни мрежи. Тъй като интересът в машинното учене не беше толкова голям, търсene на решение за този проблем не беше приоритет за много хора. За щастие обаче, една друга част от компютърната сфера, която се радва на огромен интерес от потребители, производители и иноватори има същият проблем. Става въпрос за компютърните игри, които всеки ден стават част от ежедневието на все повече хора. Проблемът пред компютърните игри е прост. Потребителите искат по-хубави графики и по-голямо количество кадри в секунда(fps). Тези две неща зависят главно от видео картата. По-специфично рендерингът на кадър се състои от голямо количество подобни изчисления за всеки пиксел на екрана. Причината тези изчисления да се извършват не на процесора, а на отделна видео карта, е че процесорът има малко количество мощни ядра, а видео картата има голямо количество по-слаби ядра. Това дава предимство на видео картата, тъй като тя може да се възползва от тези много ядра, за да постигне много високо ниво на паралелизъм, което прави голяма разлика в скоростта на ренериране на кадър. Същото нещо обаче се отнася и за изчисленията нужни за трениране и работа на невронните мрежи.

Преди няколко години се достигна критична точка в способностите на видео картите, по специфично тези на Nvidia, а именно че хардуер способен да работи сравнително бързо с невронни мрежи, вече стана достъпен за много повече хора, а не само за големи организации разполагащи с голям бюджет. Изведнъж много независими разработчици започнаха да прилагат и развиват теоретични модели, публикувани по-рано от по-големи компании като Google и от изследователи в сферата на изкуствения интелект. Днес може да видим резултата от този труд под формата на най-различни приложения използващи машинно учене, за да постигнат неща, които до скоро се смятала за научна фантастика. Примери за това са големите езични модели като GPT 3.5/4, които доста убедително репликират човешки текст, дифузните модели, като Stable Diffusion и Midjourney, които могат да генерираят изображения от зададен текст, Eleven Labs за вокализация на текст чрез имитация на човешки глас и други.

Цел и задачи

Целта на дипломната работа е да се разработи видео игра, цялата графика в която е създадена от дифузен модел, за да се тества способността на дифузните модели да изпълняват работата на графичен артист при разработката на видео игра и да се намерят ограниченията и проблемите, които това ще създаде. Също така ще се потърсят възможни решения за тези проблеми.

Задачите, които трябва да изпълним са следните:

1. Избиране на подходящ дифузен модел за генериране на изображенията.
2. Избиране на подходящ тип игра, който може добре да използва тези изображения.
3. Избиране на платформа за разработка на играта.
4. Планиране на стил, съдържание, механики и цели в играта.
5. Планиране на архитектурата на кода на играта.
6. Изучаване на начина на работа на дифузния модел.
7. Създаване на поток за обработка на изображенията създадени от дифузния модел и евентуалното им добавяне, и интеграция в играта като графични елементи.
8. Тестване и оценка на ограниченията на използваната графика.
9. Редуване на разработване и тестване на играта по планираната архитектура
10. След завършване на разработката се прави пълен тест на завършената игра.
11. Поправяне на намерените проблеми.
12. Планиране на възможни бъдещи подобрения по проекта.
13. Изготвяне на заключение за способностите на дифузния модел и за качеството на крайната игра.

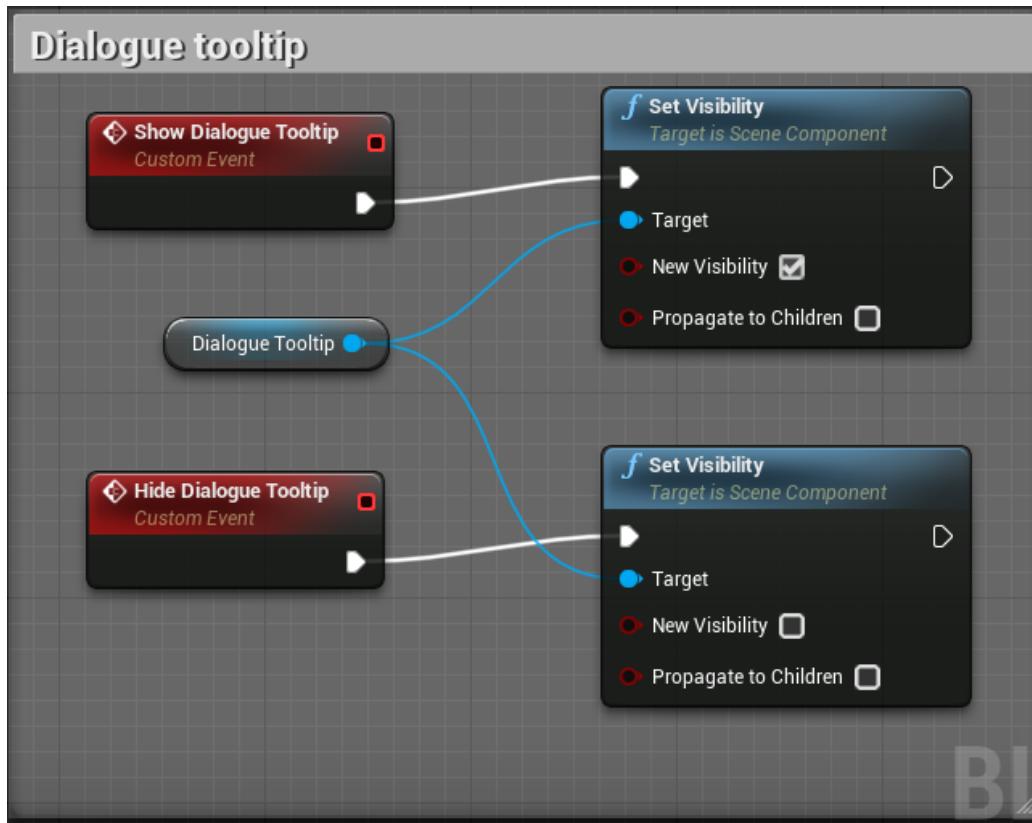
Използвани технологии

За постигане на целите и разработка на играта са нужни платформа за разработка на играта, дифузен модел за графиката, програма за обработка на резултата от модела и програма за обработка на звук. За целта съм изbral следните:

Unreal Engine 4.26

Unreal Engine 4.26 за платформа за разработка на играта. Unreal Engine е една от най-големите платформи за разработка на игри. От всички публично достъпни платформи Unreal Engine е най-използван от големите “AAA” производители. На Unreal Engine са правени много известни игри като PlayerUnknown’s Battlegrounds, Fortnite, ARK: Survival Evolved, Sea of Thieves, и много други. Също така с появата на Unreal Engine 5 много големи студии решиха да заменят собствената си платформа за разработка с Unreal Engine, например CD Projekt Red обявиха, че следващата Witcher игра ще използва Unreal Engine. Програмата предлага функционалност, която значително ускорява процеса на създаване на игра. Примери за това са готова поддръжка на файловите формати, които ще се използват за звук и графика, добър браузър на файловете, който позволява оцветяване на папки в различен цвят за по-лесно и бързо разпознаване и удобен за използване редактор за нива, който позволява правене на бързи промени по нивото и обектите в него, без да се налага бавният процес на гадаене и промяна на координати в кода, докато обекта не е на правилното място.

Платформата също предлага предварително разписани функции за по-основните механики, които се съдържат в повечето съвременни игри, като камери, прихващане на клавишно натискане и колизии. Едно от най-големите предимства на Unreal Engine е така нареченото “Blueprint” програмиране. То дава възможност на разработчика да пише кода визуално, а не като текст. На фиг. 1 може да се види как една функционалност може бързо да се свърже в един blueprint, вместо да се разписва като код. Най-близкият еквивалента на съдържанието на един blueprint в стандартния код е клас файл. Стандартен код може да се пише в Unreal Engine чрез използването на специални променливи, след което функциите в този файл с код могат да се капсулират от платформата и да се използват в blueprint по същия начин като вградените функции.



Фиг. 1: Прост пример за визуално програмиране, където имаме две функции, които са вързани за клавиш на клавиатурата и при своето изпълнение извикват функцията “Set Visibility” на обекта “Dialogue Tooltip” с различни параметри.

Разработването на игра използвайки единствено blueprint кодиране си има предимства и недостатъци. Тъй като някой от функциите, които са вградени в Unreal Engine, са направени да работят широко за повече възможни употреби те са малко по-неефективни и се изпълняват малко по-бавно от функция написана специфично за нуждата на играта. Например на фиг. 1 може да видим как “Set visibility” има вход за булева променлива “Propagate to Children” и съответната функционалност. Това е защото класът на “Dialogue Tooltip” е наследил тази функция от базовия си клас “Paper Sprite”. Друг недостатък е, че има някой ъглови ситуации в които определена функционалност не може да се репликура в blueprint. Пример за това е многомерен масив от обекти. За щастие тези ъглови случаи са малко. Предимствата на проектите, писани само на blueprint-и, е че те могат да се компилират за всички платформи, които Unreal Engine поддържа, без да се налага разработчика да променя кода, за да се

съобразява с нуждите на определена платформа. Unreal Engine може да компилира за Windows, Mac, Linux, Android, Ios, Xbox, Playstation, Nintendo Switch, различни VR платформи и други. Писането чрез blueprint-и използвайки готовите библиотеки на Unreal Engine също така отнема много по-малко време, отколкото да се пренаписват базовите класове за да са по-специфични за играта, което може да отнеме много време и ресурси. Unreal Engine също има добри инструменти за дебъгване, които помагат много с тестването на играта по време на разработка.

Остава само един въпрос. Защо Unreal Engine 4 , а не 5? Причината е, че двете версии имат различни правила относно използването на играта за комерсиални цели. При версия 5 първите \$5000 брутна печалба остават изцяло за разработчика, след което “Epic Games” изискват 5% от брутната печалба. При версия 4 обаче нещата стоят различно. Вместо \$5000 границата се премества на 1 милион долара преди правилото за 5% да влезе в сила. Това практически означава, че цялата печалба остава за разработчика освен, ако играта не стане адски популярна.

А от какво се лишава играта тъй като е на по старата версия? За щастие нищо. Подобренията в новата версия са почти ексклузивно само върху 3D игрите. Тъй като ще се използва 2D графика направена от дифузен модел, проектът не се лишава от нищо.

Midjourney

Midjourney за дифузен модел. Изборът тук е между Midjourney и Stable Diffusion. И двата варианта си имат силни и слаби страни:

- 1. Цена:** Абонаментът за Midjourney струва \$36 на месец, а Stable Diffusion е с отворен код и следователно е безплатен, ако човек разполага с достатъчно мощна видео карта, за да го подкара. Струва си да се отбележи, че има сайтове, които предлагат онлайн версия на Stable Diffusion, която работи на сървъра и може да се използва срещу заплащане на всяка отделна генерация.
- 2. Начин на употреба:** Midjourney се използва чрез популярното приложение за онлайн разговори “Discord”, и генерацията се извършва чрез изпращане на съобщение до автоматизиран бот на Midjourney започващо с “/imagine” последвано от текст описващ изображението, което искаме да генерира. Като резултат започват да се генерират 4 изображения и този процес може да се наблюдава. След завършване на

генерацията, ако резултатът не ни хареса има копче за повторна генерация, което ще генерира 4 нови изображения, а ако ни хареса може или да кажем на генератора да направи още 4 изображения, които са подобни на едно от първите 4, или да изберем едно от тях и да накараме генератора да го отдели от другите 3 в собствено изображение. Във по-старите модели разделянето също прави “upscale” на изображението, което го прави с по висока резолюция, но в новата версия на модела изображенията директно се генерират с по-високата резолюция. Midjourney също има настройка до колко дифузерът да се придържа към текста, който му е даден. След генериране изображенията също могат да се достъпят на сайта на Midjourney.

Stable Diffusion от друга страна има много начина за използване. Тъй като е с отворен код, има много различни интерфейси, които могат да се изтеглят от места като Github и съответно имат различни интерфейси и начини на ползване. Също така може да се използва един от гореспоменатите сайтове, срещу заплащане. След като се избере начин на достъп до дифузера трябва да се избере и модел, който представлява “знанията”, които дифузерът ще използва. Различните модели дават различни резултати. Някои са тренирани на голямо количество картини и снимки на малко количество обекти, което ги прави по-добри в генерирането на картини с тези обекти, но по-слаби във всичко останало, но има и такива които са тренирани на много обекти, които рисуват с по-малък детайл. От страна на настройките Stable Diffusion е далеч по-modифицируем от Midjourney, но това също означава, че изисква повече знания за да се използва оптимално.

3. **Скорост на генерация:** Скоростта на генериране на 4 изображения с Midjourney варира между 10-30 секунди с приоритет и между 10 секунди до 2-3 минути без приоритет когато сървърът е по-натоварен. Могат да се слагат на опашка до 15 заявки за генерация и се обработват по 3 наведнъж. За Stable Diffusion времето за генерация зависи от мощността на видео картата, която се използва, броят итерации и резолюцията зададени в настройките. Със RTX 3060 12GB отнема около 6-7 секунди за генериране на изображение с 30 итерации и резолюция 512x512.

4. Качество на генерацията: На фиг. 2 може да се види разликата между генерацията на Stable Diffusion и Midjourney. Качеството на генерация на Stable Diffusion много зависи от използваният модел. Със стандартният 2.1 модел няма голямо ниво на реализъм или на точно определен стил, а при Midjourney има много по-добро осветление, което изглежда натурално, гладко преливане на цветовете и най-важното е, че е много по-близо до текста, който е даден за генерация



Фиг. 2: Разлика между генерациите на Stable Diffusion(ляво) и Midjourney(дясно) при зададен текст - A magnificent crow eating purple grapes in the throne room of a castle

GIMP

GIMP за графичен редактор за обработка на резултата от генератора на изображения. GIMP е идеален редактор за целта на проекта. Той има далеч повече способности от стандартният редактор на Windows и е с отворен код, което го прави по-привлекателен от програма като Photoshop, която е скъпа. Въпреки, че няма всички способности на Photoshop, GIMP има достатъчно инструменти, за да може да направи генерираните резултати използвани за играта. Основните компоненти, които ни трябват са възможност за създаване на прозрачен фон, умни инструменти за избиране, високо ниво на приближаване и прецизни инструменти за редактиране на ниво пиксел.

Sony Vegas Pro 15

Sony Vegas Pro 15 за звуков редактор. Този избор не е особено практичен, ако разработчика не е предварително запознат с програмата. За бесплатна и по-практична алтернатива може да се използва програма като Audacity. Звуковият редактор ще е нужен за обработка на звуци, които да се сложат в играта.

Архитектура

Преди да започне разработката трябва да се определи размера и типа на играта, функционалността, която трябва да е налична и начина, по който различните обекти ще взаимодейства един с друг.

Тип

Типът на играта ще е 2D Sidescroller. В този тип игра играчът управлява герой, който гледа от страни, и може да го движи наляво, надясно и в повечето случаи може да скача. Целта е да се преминат успешно всички препятствия и да се достигне до края на нивото. Известни игри от този тип са старите Mario и Sonic игри. Тъй като целта на дипломната работа е да се тестват графиките, направени от дифузния модел, то е много по-практично да се направи 2D игра в която цялата графика се състоеи от 2D картички. Ако играта е 3D, то ще трябва да се използват картинките върху 3D модел. Това е възможно за нещо като материал напр. трева или скали, но за момента дифузера не може да прецени как да направи картичка на човек, която да се сложи на 3D модел. Друго предимство на 2D игрите е, че те в повечето случаи са по-бързи за разработка, тъй като се елиминира нуждата от правене на 3D модели.

Размер

Размерът на играта ще се цели към 25-50 минути игрово време за нов играч от начало до край. За да се постигне това, играта ще има 8 нива, като първото ще представлява лесно начално ниво, което да даде шанс на играта да свикне с контролите на играта, а последното ще бъде трудно ниво, което най-вероятно ще отнеме няколко опита преди да бъде преминато успешно. Трябва да има разнообразни препятствия и противници, за да може играчът постоянно да среща нещо ново във всяко ниво.

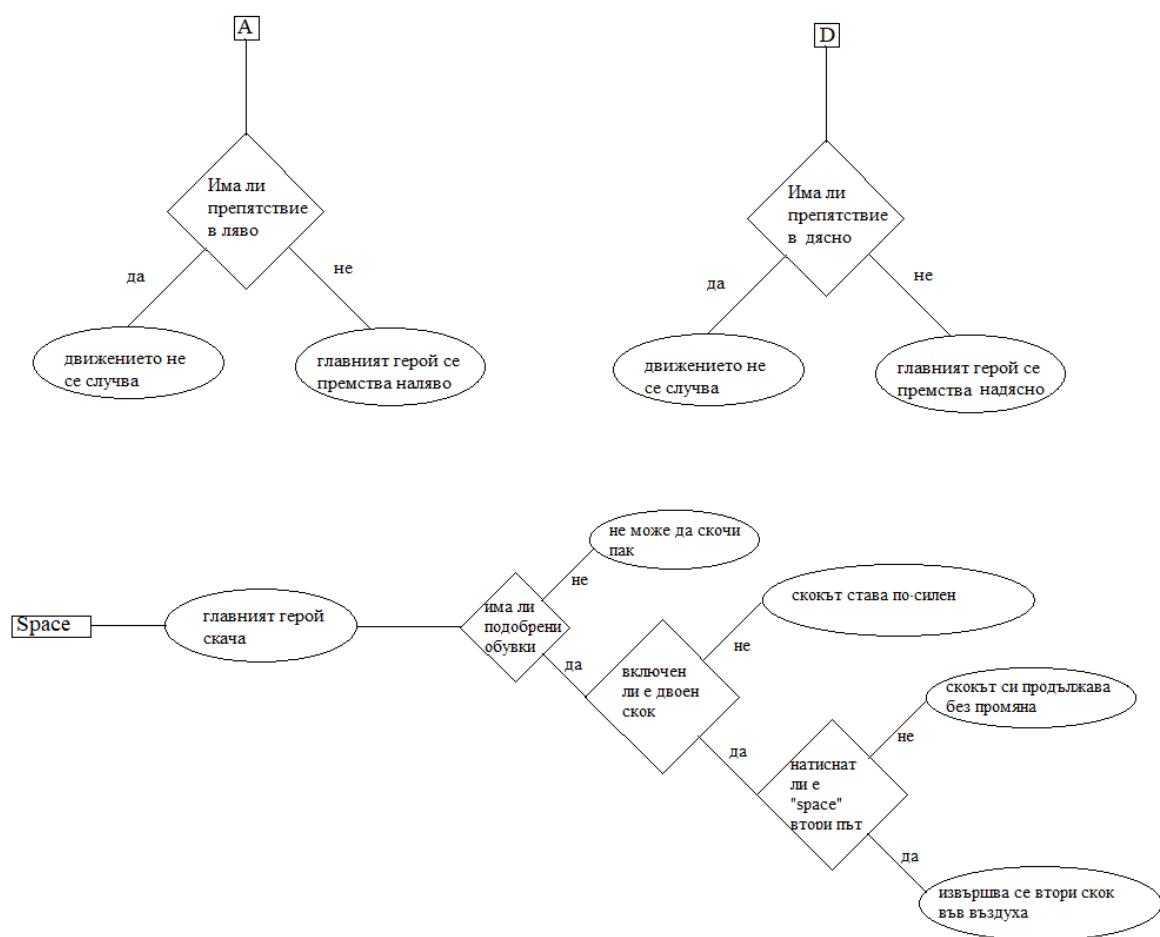
Основна Функционалност

Система за движение на главния герой

В фиг. 3 може да се види как системата за движение на главния герой работи. При натискане на "A" героят се опитва да се движи наляво, но ако има препятствие

движението не трябва да се случва. Същото се отнася за копчето “D”, но този път посоката на движение е надясно. С натискане на “Space” се извършва стандартен скок. Този скок ще може да се подобри по време на играта, което ще даде възможност за втори въздушен скок. Този скок ще може да се направи по всяко време докато героят е във въздуха и ще добави още вертикална сила нагоре. Ако героят пада в момента на втория скок, то вертикалната сила надолу се занулява преди да се добави повдигащата сила. По-късно в играта героят ще има достъп до втори режим на скок, който ще замени двойния скок със един по-силен скок.

Система за движение



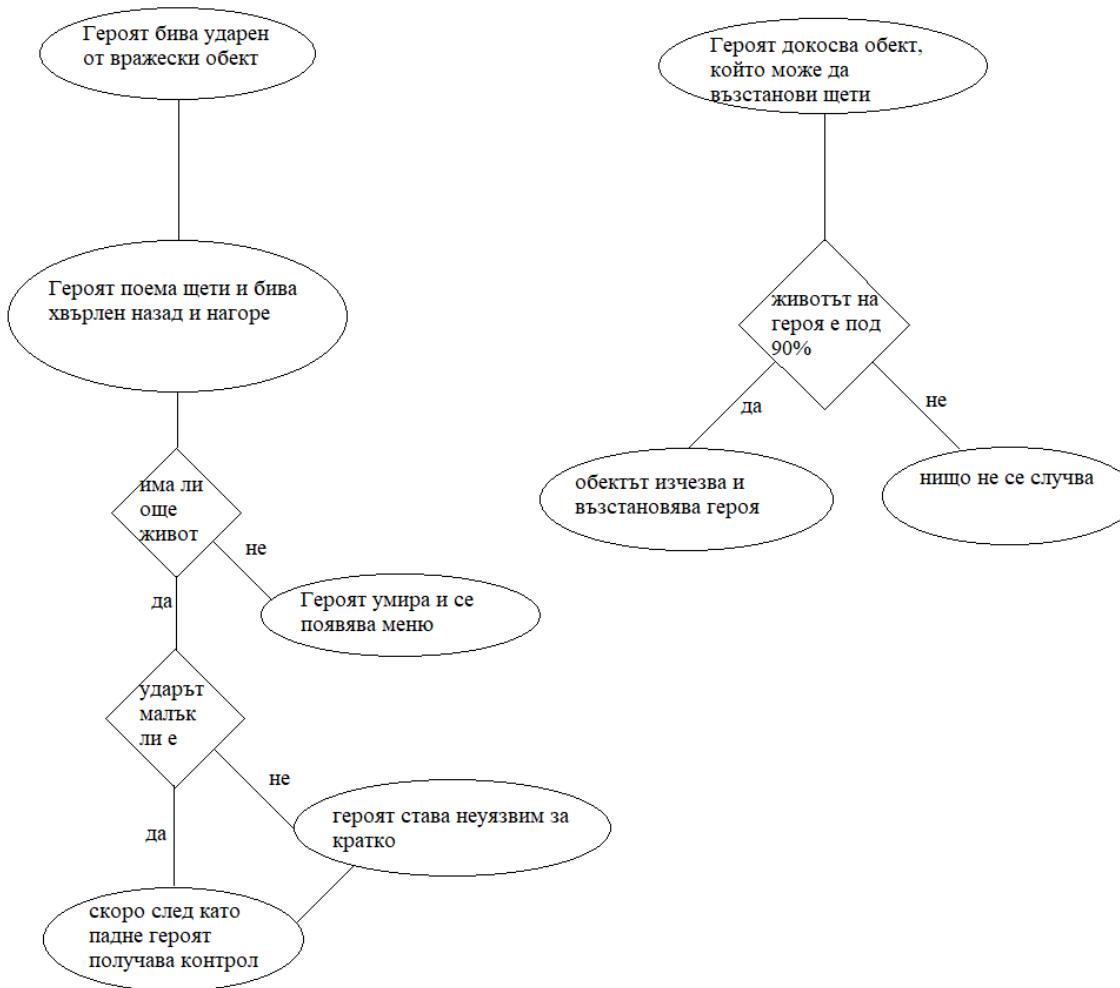
Фиг 3: Диаграма показваща системата за движение на главния герой

Вторичната система за движение идва от околната среда. Това са неща като препятствия, врагове и други видове терен. Например, ако главният герой стъпи на трън, то той ще бъде бутнат назад и нагоре, а след падане на земята, ще му отнеме малко време да се възстанови преди да може да се движи отново. Друг елемент на тази вторична система е интеракцията на героя с терен. Когато той стъпи на нещо, което се движи и може да остане на него, то героят трябва да се движи заедно с това нещо. Когато героят скочи наляво или надясно, то тази инерция от обекта, на който стои, трябва да се запази и ако скокът е в същата посока, то инерцията трябва да се добави към хоризонталната скорост на скока.

Система за поемане и възстановяване на щети

Поемането на щети ще може да стане по два различни начина. Първият ще е чрез колизия с враг или вражески обект. В този случай героят ще се изхвърли назад и нагоре и ще поеме количество щети в зависимост от обекта. Някой по-големи обекти също дават на героя кратка неуязвимост. Вторият начин ще е чрез стъпване в капан или колизия с някои видове терен. Отново ще има отблъскване и поемане на щети, но тук не се дава неуязвимост на героя. Този процес може да се види на фиг. 4.

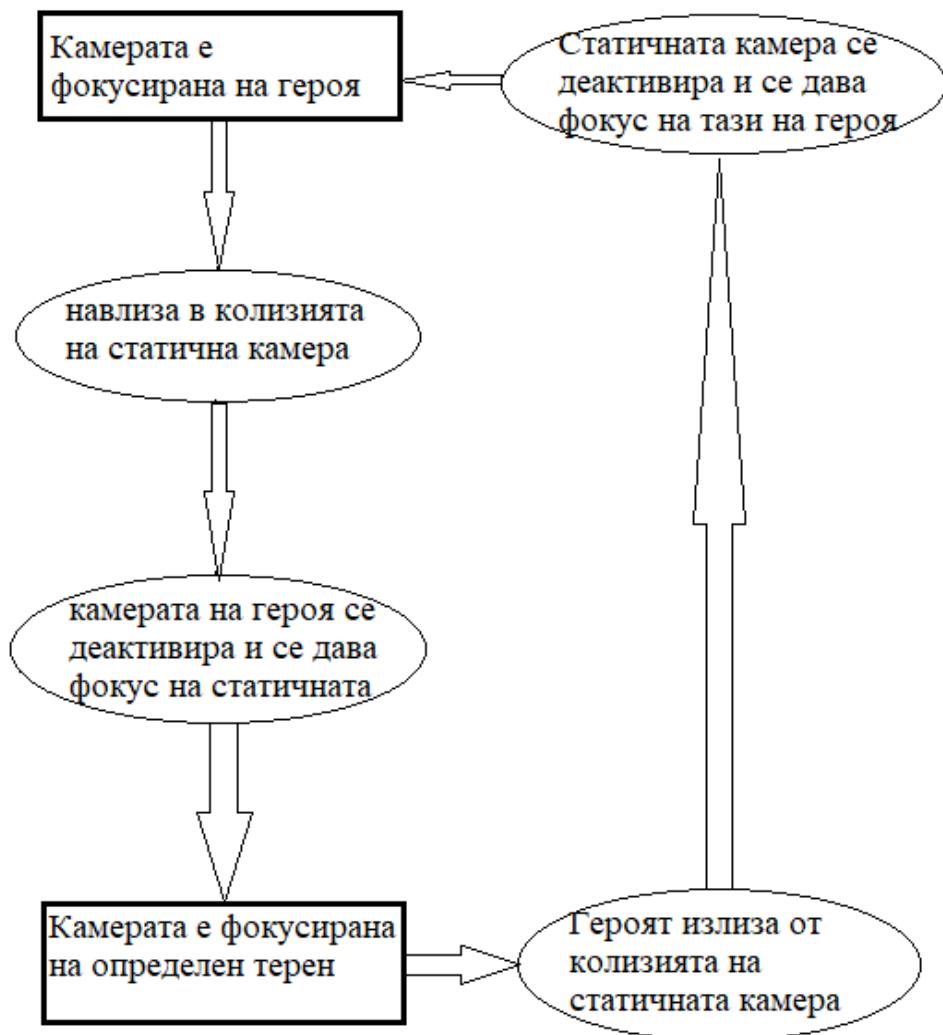
Процеса на възстановяване на щети също може да се види на фиг. 4. При контакт с обект, който има способността да възстанови щети, ще се прави проверка дали героят е достатъчно ранен преди да се консумира обектът. Ако условието е изпълнено, то обекта ще се консумира и ще възстановява щети на героя. В противен случай обектът ще си остане на мястото за евентуална по-късна консумация.



Фиг. 4: Диаграма на системата за щети

Система за динамична камера

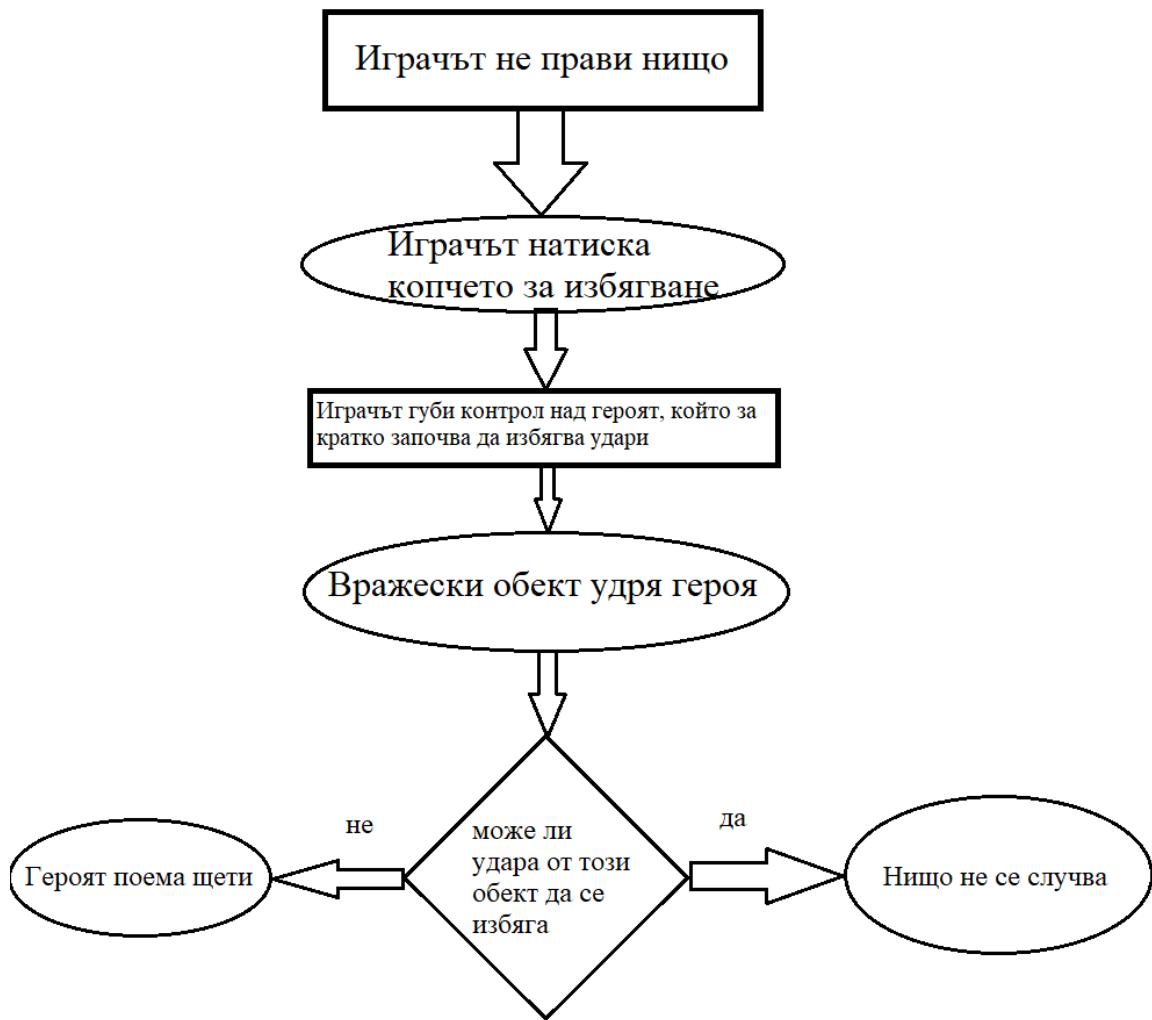
Целта на системата за контрол на камерата е да подобри видимостта на определени препятствия и да направи части от нивото да се усещат като едно цяло. Това ще се постигне чрез добавяне на статични камери на определени места из нивото. Когато играчът навлезе в зададената колизия на тази камера, то камерата, която се движи с героя ще се деактивира и ще се подава фокус на статичната камера. Тази промяна ще се извърши плавно между координатите на героя и координатите на статичната камера. При излизане от колизията този процес ще се извърши обратно и камерата на играча отново ще става тази, която е в фокус. Този процес може да се види на фиг. 5



Фиг. 5: Схема на системата за смяна на камери

Система за избягване на определени атаки

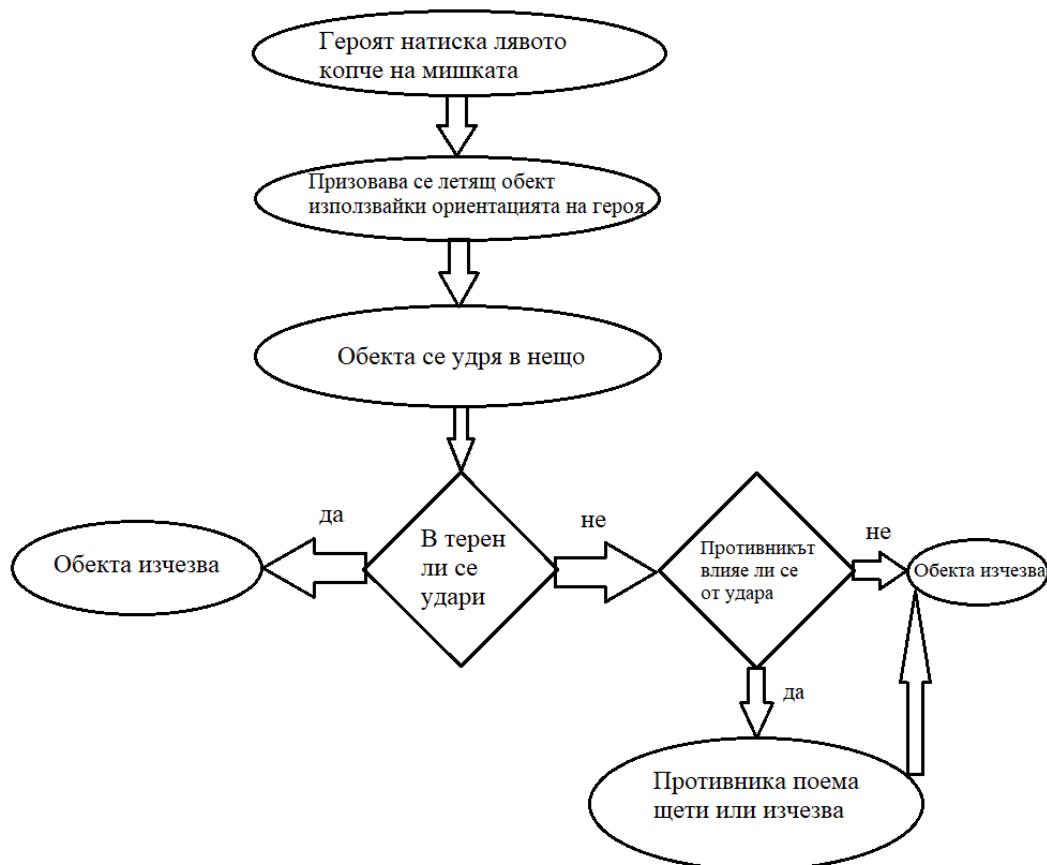
Тази ще дава възможност на играта да избяга определени атаки. Начинът на работа може да се види на фиг. 6. При колизия с голям вражески обект избягването ще е невалидно и героят поема щети по стандартния начин. При колизия с обект хвърлен от противник, избягването ще е валидно и ще предотврати всякакви щети. Важно е също да се отбележи, че играчът губи контрол по време на избягването.



Фиг. 6: Система за избягване

Система за хвърляне на обекти

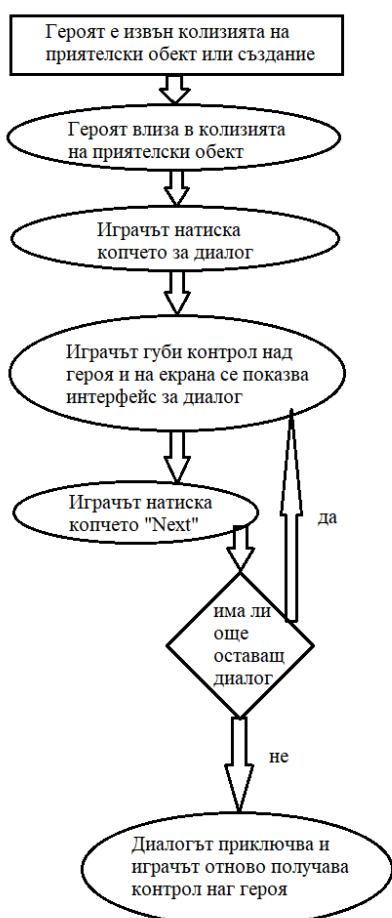
Целта на тази система е да даде възможност на играчът да хвърля обекти, които могат да нанесат щети на враг или напълно да го премахнат от нивото. Това умение ще се използва за последното ниво на играта. Ще има максимално количество обекти за хвърляне, които героят може да носи и съответно ще има начин да се сдобие с още. При хвърляне на обекта, той ще лети в права линия, без да се влияе от гравитация. При колизия с противник обектът ще изчезва, а удареният противник ще получава сигнал, че е ударен. В зависимост от вида на противникът, той ще реагира различно. Големи противници няма да се влияят от удара, а малките или ще поемат щети, или директно ще бъдат махнати от нивото. Начина на действие на тази система може да се види на фиг. 7.



Фиг. 7: Система за хвърляне на обекти

Система за диалог

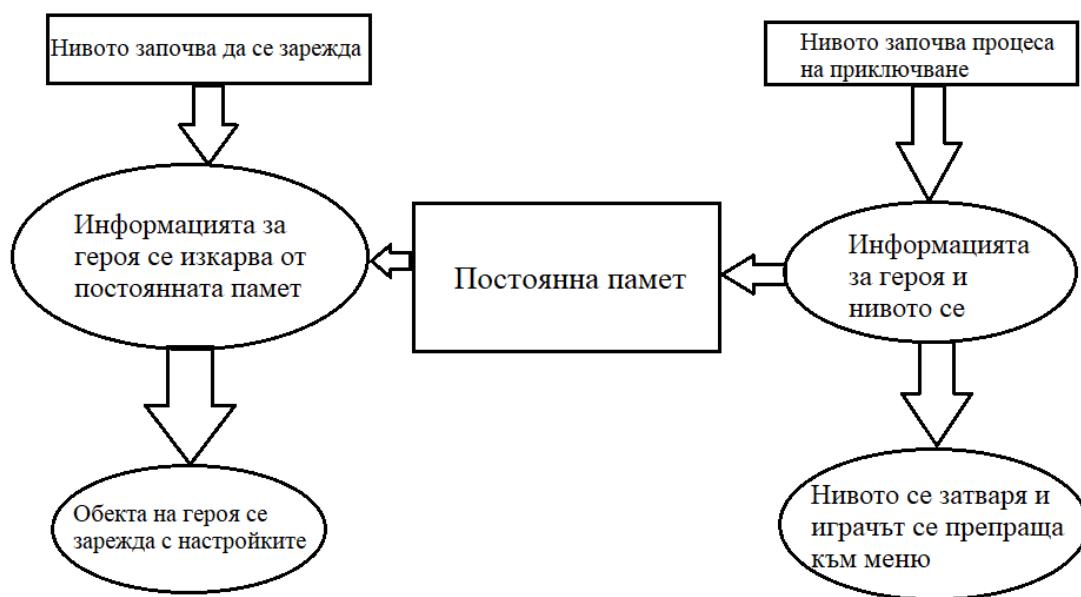
Целта на системата за диалог е да подпомогне разказването на историята на играта. Като кого играя? Какъв е света, в който се намира играта? Защо минавам тези нива? Как получих това подобрение? Това са въпросите, на които ще се отговори чрез системата за диалог. Тя също ще позволи разговор с приятелски създания. Системата ще се състои от 2 основни компонента. Първият е приятелско създание или обект, което ще се поставя в нивото, за да може героят да взаимодейства с него. Вторият ще е интерфейс, чрез който ще се извършва диалог. На фиг. 8 може да се види процеса. Чрез тази система също така ще се дават подобрения на героя, ще се премахват или поставят невидими стени из нивото и други различни действия свързани с темата на разговора. Ще има версия на интерфейса, която е праволинейна(показаната на фиг. 8) и такава, която дава различни опции за продължаване на разговора.



Фиг. 8: Система за диалог

Система за запазване на прогрес

Тази система ще е отговорна за запазването на прогреса на играча между различни сесии. Тя ще трябва да запазва всички отключени нива и подобрения, които играчът е постигнал. Системата трябва да работи автоматично без да занимава играча. За да се постигне това, на края на всяко ниво системата ще се активира и ще запазва прогреса. Системата ще се използва и в началото на всяко ниво за да придае нужната информация за отключени подобрения на обекта на героя.

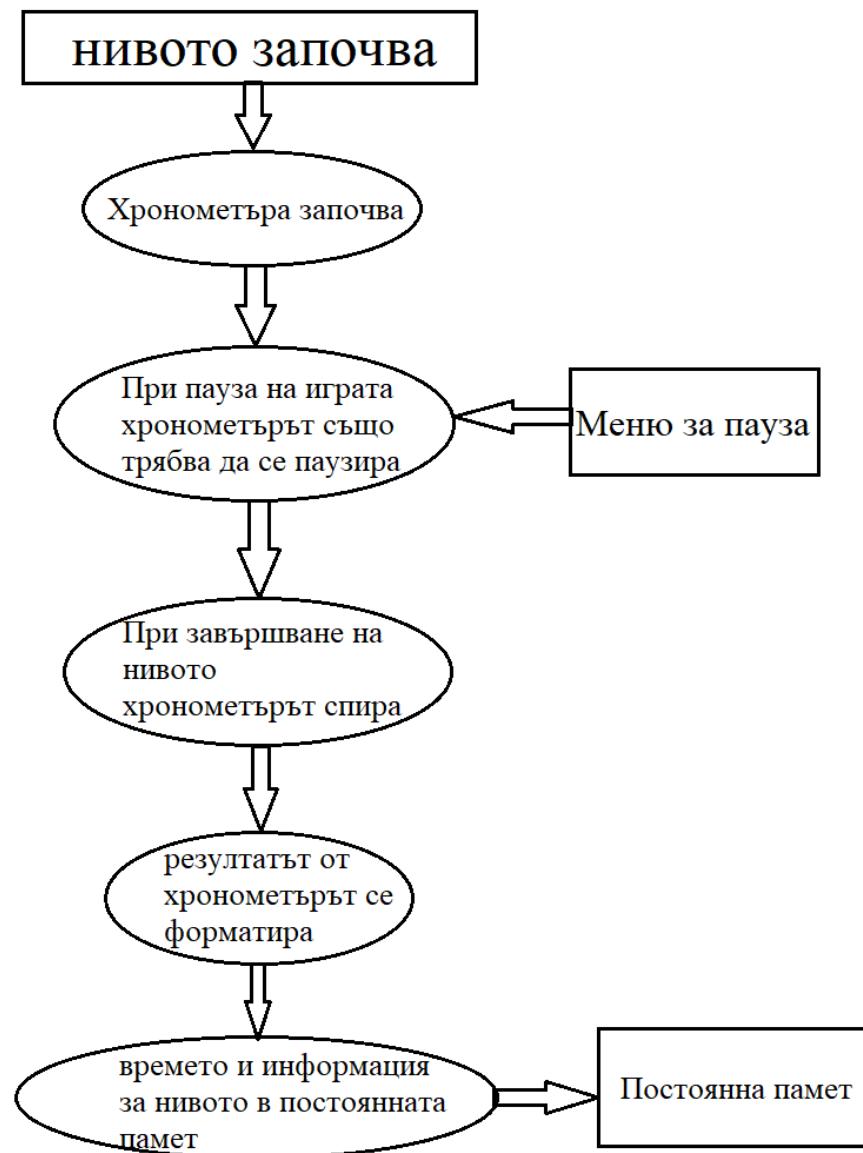


Фиг. 9: Работа на системата за запазване

Система за запазване на рекордно време

Целта на тази система е да запазва рекордното време, за което всяко ниво е минато. Това време е нужно, за да може да се направи един изглед с най-бързото време за всяко ниво. Подобни неща привличат група от играчи, които обичат да се състезават помежду си. Системата ще съдържа два компонента. Първият е хронометър. Той трябва да се стартира при начало на нивото и да записва времето с точност до милисекунда. Също така трябва да може да се поставя на пауза, за да не се влияе от отваряне на менюта. Другата част е подобна на системата за запазване на прогрес, но е специализирана само

за запазване на време. При края на нивото хронометърът трябва да спре, да форматира времето в четим за човек формат и да го изпрати към компонента за запазване, заедно с информация за нивото.



Фиг 10: Система за запазване на най-бързо бреме

Архитектура на менютата

В играта ще се използват следните менюта:

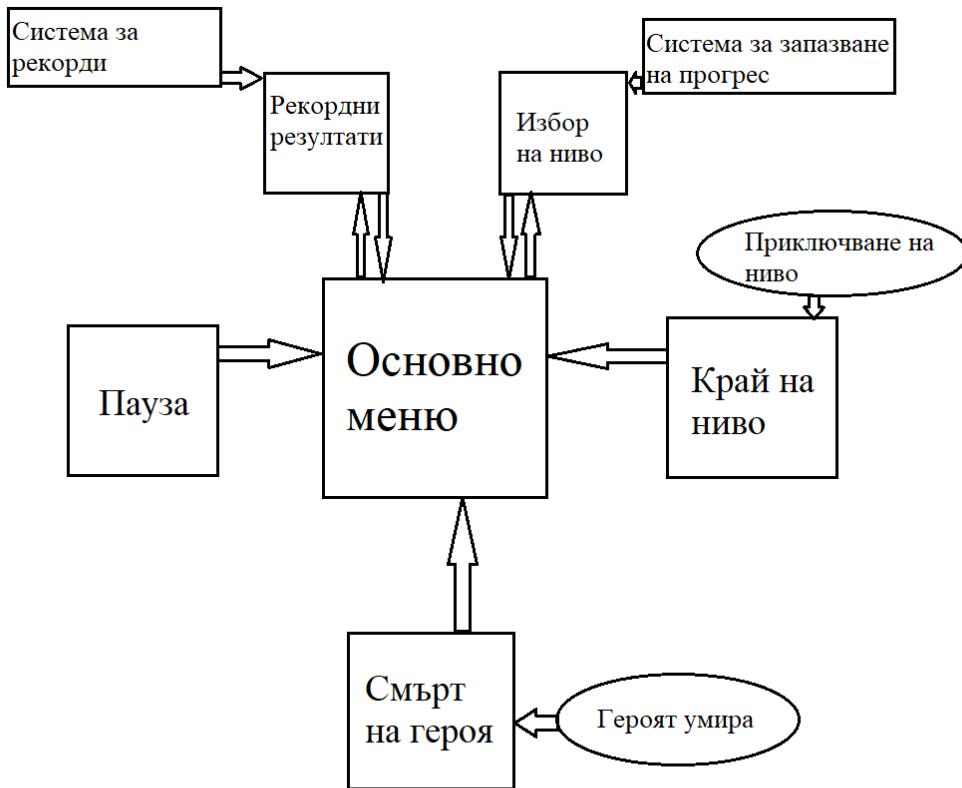
Основно меню - в ще има копче за начало или продължаване на играта, копче за избор на ниво, копче което отваря изглед с рекордните резултати, копче за изчистване на прогрес и копче за излизане от играта.

Меню за пауза - в него ще има копче за продължаване на игра, копче за връщане в основното меню и копче за излизане от играта

Меню за край на ниво - в него ще има копче за следващо ниво, копче за връщане в основното меню и копче за излизане от играта

Меню при смърт на героя - в него ще има копче за рестартиране на сегашното ниво, копче за връщане в основното меню и копче за излизане от играта

Структурата и връзките между менютата се виждат в фиг. 11.



Фиг. 11: Структура на менютата

Обекти и взаимодействия

За целта на играта ще са нужни следните **обекти и взаимодействия между тях**:

1. **Главен герой** - основният клас, чрез който играчът ще взаимодейства на играта. Тук ще се съдържат голяма част от системите на играта.

2. **Вражески същества** - Те ще представляват опасност за главният герой. Различните същества трябва да атакуват по различен начин.
3. **Вражески капани** - Те също ще са опасност, но за разлика от вражеските същества, те няма да могат да се движат или да удрят героя без негова намеса.
4. Вражески обект, който ще се хвърля от вражеско същество.
5. Обект, който героят ще хвърля.
6. Обект, който ще възстановява щети на главния герой.
7. Статична камера, която да се поставя в нивото.
8. Обект или терен, който ще представлява край на нивото.
9. Невидима стена, която да ограничава достъпа до определени места.
10. Приятелски същества и обекти, с който играчът ще може да говори.

Разработка на играта

След завършване на планирането на архитектурата на играта вече може да започне разработката.

Процес за създаване на графиката

Преди да започне разработката на системите и нивата трябва да се измисли процес, който да започва с генерация на изображение на обект от дифузния модел - Midjourney, след което да се обработва в GIMP и накрая да се внедрява в Unreal Engine.

Етап I: Генерация

За генерация на изображения се използва Midjourney. Тук има няколко важни момента, на които трябва да се обърне внимание. Първо е важно да се постигне максимално подобен стил между различните генерации. Това е важно, за да може отделните обекти да не изглеждат твърде различни един от друг. За да се постигне това, трябва да се спазват две правила. Едното правило е да се използва един и същи модел при генерацията на всички изображения. За играта се използва модел 5.1 на Midjourney. Другото правило е да се измислят няколко стилизиращи думи, които се поставят на края на всяка генерация, след описанието на нужния обект. За играта тези думи са следните: “cartoon style, hand-drawn, cardboard-texture, moody colors, medieval, rustic”. Например, ако трябва да се генерира сив камък, вместо да се даде команда “/imagine gray stone”, се дава команда “/imagine gray stone, cartoon style, hand-drawn, cardboard-texture, moody colors, medieval, rustic”. Това се прави, за да повиши приликата между различните обекти.

Вторият важен момент е при генериране, разглеждане и избиране на изображения, който да продължат в следващия етап. Първо при генериране на голямо количество изображения, което ще е нужно при повечето обекти, е по-добре да се използва така нареченият “relaxed mode” на Midjourney. За разлика от “fast mode”, той не дава по-голям приоритет на изпълнение на командата, която е изпратена, пред други потребители, което води до по-бавно начало на генерацията. Предимството обаче, е че имам неограничен брой бавни генерации включени в абонамента, а бързите са ограничени и е по-добре да се използват за определени генерации, които трябват по-спешно. Това по-бавно генериране не е голям проблем тъй като могат да се поставят

на опашка до 15 команди на веднъж и докато се генерираят може да се работи по кода на играта. След като се генерираят може да се разгледат на веднъж и да се отделят тези, които отговарят на следните условия за да се улесни следващият етап:

- Изображението има сравнително празен фон зад обекта, който ни трябва. За да се увеличи шанса за този вид фон се слагат аргументи като “empty background” или “no-background”
- Изображението близко наподобява стила на другите обекти, които вече са готови
- Обекта не е видимо деформиран или тези деформации могат лесно да се оправят в следващият етап
- Желателно е обекта да има поне малко очертание разделящо го от фона. Това се постига чрез аргументи като “outline” и “thick-outline”. Това ще помогне в следващият етап.
- Обектът изглежда естетически добре и няма елементи, които го правят труден за интегриране в играта.
- Обектът не изисква големи промени за да бъде завършен(например скала която се е генерирала с гущер на нея, който е нежелан). Това условие може да се игнорира ако разработчика има опит с дигитално рисуване.

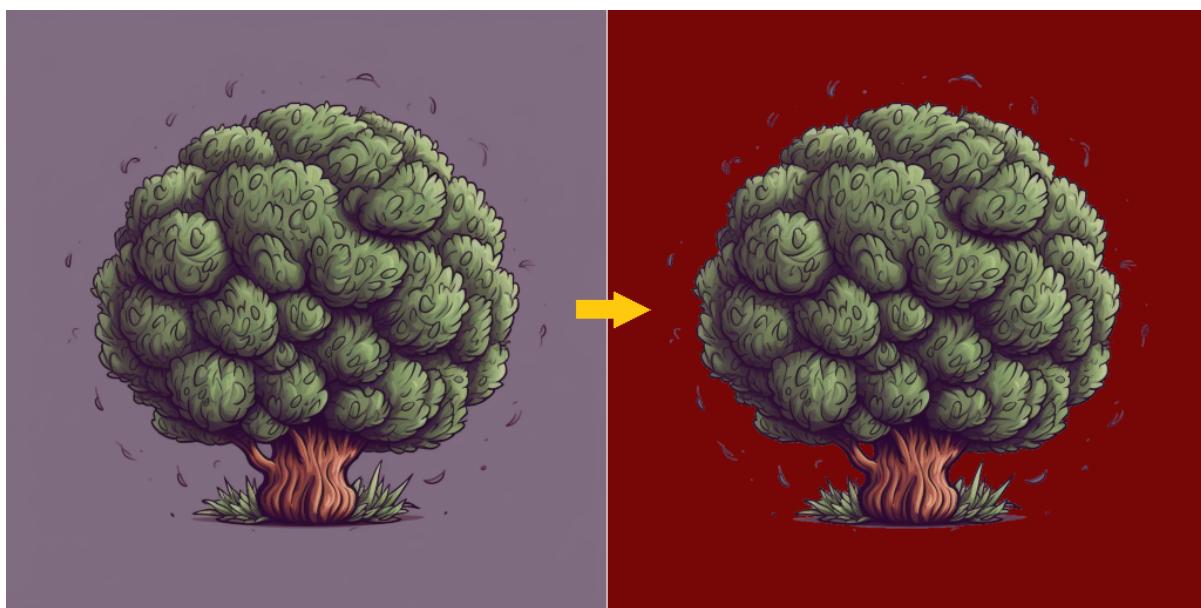
Изображенията, които успешно отговарят на тези условия се отделят и се теглят от сайта на Midjourney. След това се преминава към етап две

Етап II: Обработка с GIMP

В този етап вече има подбрани изображения съдържащи обектите, които ни трябват за играта и сега те трябва да се отделят от фона и да се направят нужните корекции по тях. Първо се отваря изображението в GIMP и се навигира до менюто за слоеве в долният десен ъгъл. С натискане на дясното копче върху слоя, на който е изображението, се отваря падащ списък, в който се избира “Add alpha channel”. Това дава възможност на изображението да бъде прозрачно. След това се изпълняват следните стъпки:

Стъпка 1

Преди да започне обработката е препоръчително да се сложи още 1 слой в изображението, който да е пълно оцветен с контрастен цвят под слоя на изображението. Това се прави за по-лесно забелязване на дефекти. След това започва обработка по следния начин. Ако фонът е цвят, който не се среща в обекта, се избира инструмента “color select” и се настройва слайдера на “threshold” в ляво на около 20. След това се избира фона и се натиска копчето “delete” на клавиатурата. Ако цветът на фона се съдържа в обекта първо трябва да се провери дали очертанието напълно изолира обекта от фона и ако това не е така, то очертанието ръчно се запълва. След това се избира инструмента “fuzzy select” и отново се настройва “threshold” да е около 20. След това се избира фона и след проверка, че не е избрана и част от обекта се натиска “delete”.(Забележка: Ако фона не е еднотипен, то “threshold” се увеличава за да има по-голям толеранс към разлики. При такъв фон често се налага по-голяма степен на ръчно изтриване като стъпка 3)



Фиг. 12: При първата стъпка фонът е премахнат и е сложен контрастен цвят зад изображението

Стъпка 2

След стъпка 1 вече би трявало единственият остатък от фона да е по очертанието на обекта. Сега се избира отново един от инструментите за избор от стъпка 1 и се маркира прозрачната част на изображението. След това се натиска дясно копче в маркираният

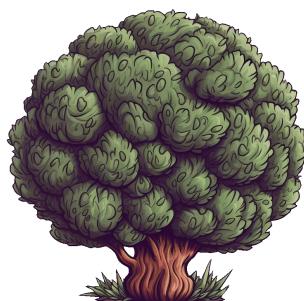
район и от падащото меню се избира “Select” и след това “Grow” и се слага за стойност 1 или 2 в зависимост колко е плътно очертанието. Това ще уголеми маркираният район и ще прихване голяма част от остатъците от фона. След това се натиска “delete”, за да се изчисти маркираното.



Фиг. 13: Изображението след стъпка 2

Стъпка 3

Накрая ръчно трябва да се изчистят всички остатъци от фона. Това се прави чрез гумичка с размер 1 пиксел и приближаване на изображението между 800% и 1600%. След като се изчисти се маха слоя с контрастен фон, ако е бил сложен, и изображението се експортира в .png формат.



Фиг. 14: Финалният резултат след стъпка 3

Етап III: Интеграция на графиките в Unreal Engine

За да се интегрира едно изображение в Unreal Engine трябва да се изпълнят следните стъпки.

1. Отваря се Unreal Engine и се навигира до папката на проекта, в която ще слагат изображенията.
2. Извън Unreal Engine се отваря папката в която е изображението
3. Изображението се хваща и се влачи вътре в прозореца на Unreal Engine и се пуска в желаната папка на проекта.
4. След това се натиска дясно копче на новопреместеното изображение и се избира опцията “Apply Paper 2D Settings”. Това настройва изображението, за да изглежда остро и да не прелива цветове. След това отново с десен клик се избира опцията “Create Sprite”. Това създава използваем графичен обект, който вече може да се използва като графика за играта.

Разработка на системите на играта

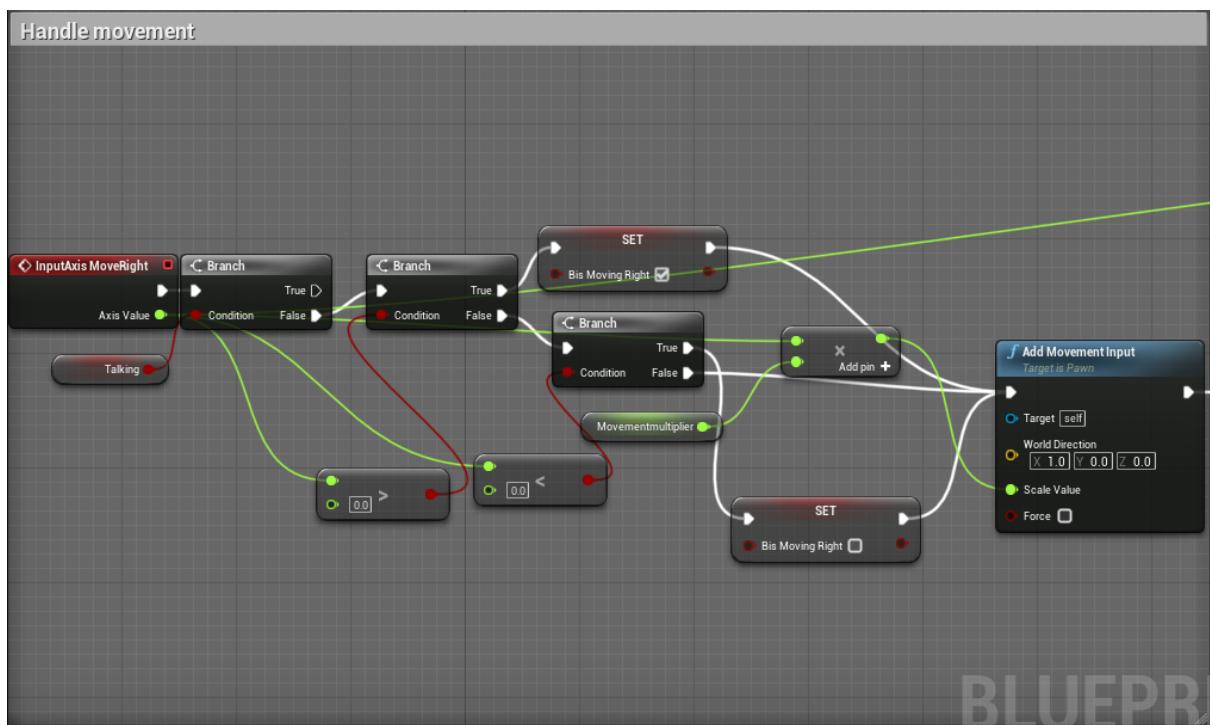
След като има измислен процес за обработка на изображенията вече може да се започне разработката на системите на играта. След създаване на Unreal Engine проект първо се създава blueprint, който наследява базовият клас “2D Paper Character” и се именува “BP_main”. Този blueprint служи като основен интерфейс между играчът и играта и съдържа функционалността на главния герой. След като е създаден може да се започне разработката на основните системи планирани в архитектурата.

Система за движение на главния герой

Първата основна система позволява на главния герой да се движи из нивата и е нужна, за да могат да се тестват другите системи. По тази причина тя се разработва първа. Тъй като е наследен класа “2D Paper Character”, вече имаме компонент наречен “Character Movement”. Менюто за компоненти се намира в горният ляв ъгъл когато отворим “BP_main”. Първото нещо което трябва да се направи е да се кликне върху компонента за движение и той да се настрой. След ляв клик върху “Character Movement” в дясната част на екрана се отваря меню “Details”, където трябва да се променят следните неща:

- Стойността на “Gravity Scale” се променя на 3.5
- Стойността на “Max Acceleration” се променя на 10000
- Стойността на “Max Step Height” се променя на 10
- Стойността на “Ground Friction” се променя на 3
- Стойността на “Max Walk Speed” се променя на 2000
- Стойността на “Braking Deceleration Walking” се променя на 10000
- Стойността на “Jump Z Velocity” се променя на 1300
- Стойността на “Air Control” се променя на 0.8
- Стойността на “Constrain To Plane” се променя на истина

След като тези настройките са зададени, трябва да се настрой прихващането на клавишите от клавиатурата, които са нужни. Това се случва чрез минимизиране на BP_main и от падащото меню “Edit” на основния прозорец на Unreal Engine се избира опцията “Project Settings” и в ново отворения прозорец се избира менюто “Input”. Вътре в менюто се създава нов “Axis mapping” и се именува “MoveRight”. След това за него се създават два бутона, които да прихваща. Това са “A” със стойност на “scale” -1 и “D” със стойност 1. Вече тези бутони ще се прихващат и може да се използват в BP_main

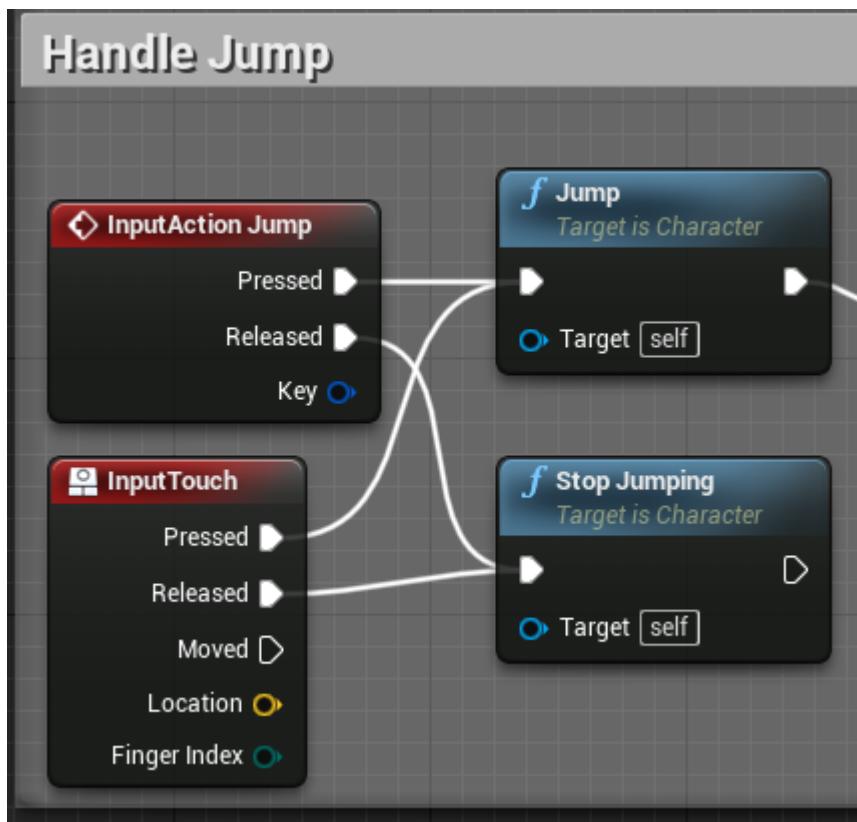


Фиг. 15: Част от кода за движение наляво и надясно

Ще са нужни и променливите “Bis Moving Right” от тип “boolean”, “MovementMultiplier” от тип “float” и “Facing” от тип integer. Те се създават от менюто

“Variables” в лявата част на BP_main. След като са създадени им се задават стойности по подразбиране “true” и 0.5 съответно. Следва разписване на кода за ляво и дясно движение както частично е показано на фиг. 15. Първо с дясно копче на мишката върху празния фон в средата на “BP_main” и се поставя “InputAxis Move Right”, който беше дефиниран по-рано. Всеки път когато играчът натисне “A” или “D” този елемент подава сигнал за изпълнение заедно със стойността на “scale”. От тук чрез проверка на стойността на “scale” ще се зададе стойността на “Bis Moving Right” и чрез функцията “Add Movement Input” на “Character Movement” ще се подаде импулс за движение в правилната посока. След това трябва да се изчисли посоката, към която героят трябва да гледа при движение и тази стойност да се запази в променливата “Facing”.

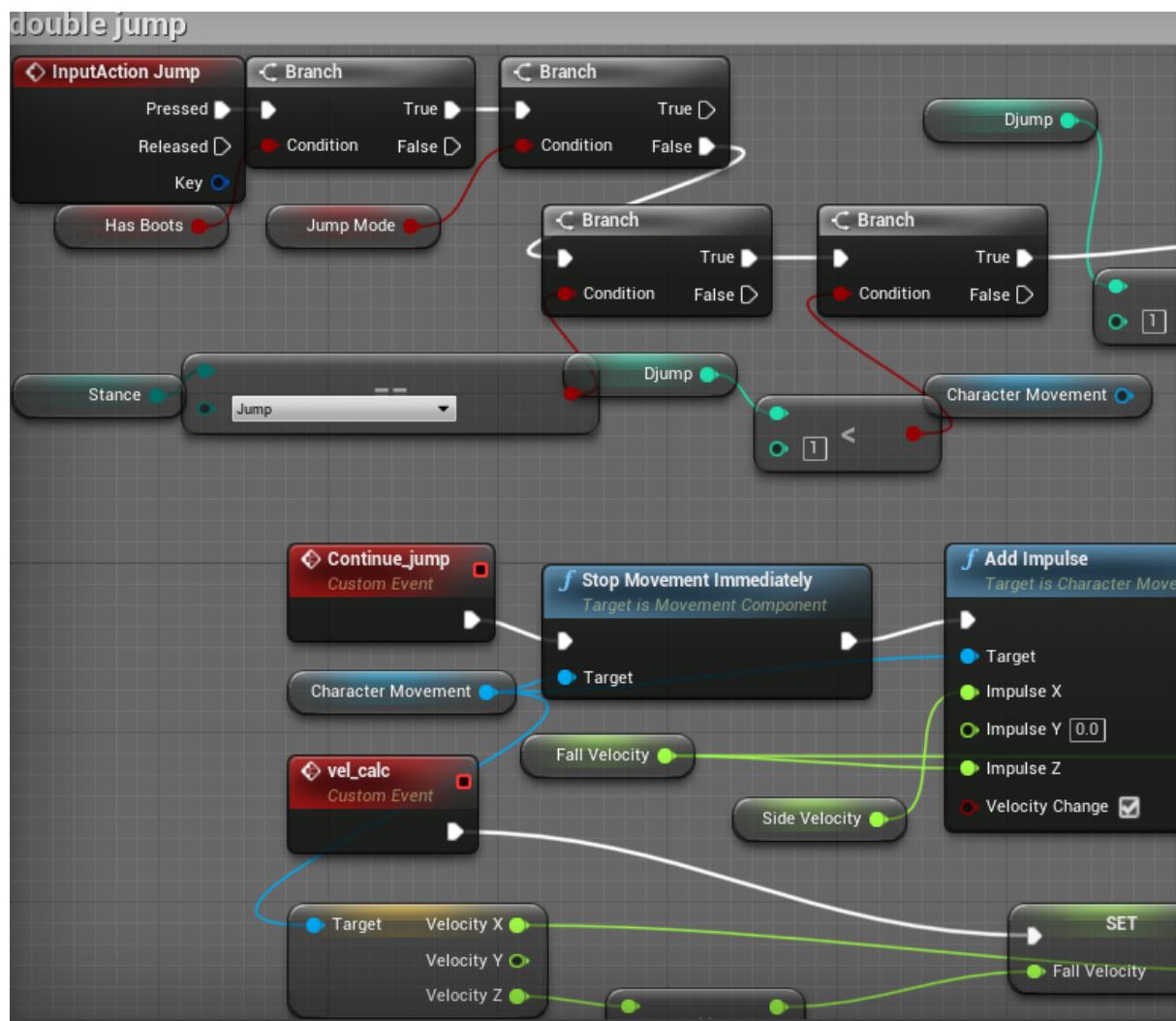
След като е готово движението настрани е време да се имплементира простото скачане. За щастие това става лесно. Първо трябва да се дефинира “Action Mapping” в настройките на проекта и ще му се зададе “space” като клавиш. След това както е показано на фиг. 16 се свързва функцията “Jump” на “Character Movement” при натискане на клавиша и функцията “Stop Jumping” при край на натискане на клавиша.



Фиг 16: Имплементация на скок

След като са готови основните движения е време да се направи функционалността за подобрените скокове. Първо ще са нужни следните променливи:

- Променлива “Has_boots” от тип “boolean” със стойност по подразбиране “false”
- Променлива “jumpMode” от тип “boolean” със стойност по подразбиране “false”
- Променлива “upgradedBoots” от тип “boolean” със стойност по подразбиране “false”
- Променлива “djump” от тип “integer” със стойност по подразбиране 1
- Променлива “fall_velocity” от тип “float” със стойност по подразбиране 0
- Променлива “Side_velocity” от тип “float” със стойност по подразбиране 0



Фиг. 17: Част от кода на двойния скок където се правят проверки

Булевите променливи се използват за отключване на подобренията на скока. На фиг. 17 са показани част от проверките, които се правят преди да се продължи до кода за двоен скок. С променливата “djump” се задава броя позволени допълнителни скокове. Тази проверка се извършва при повторно натискане на “space” докато героят е във въздуха. Ако проверката е успешна то се запазват стойностите на хоризонтално и вертикално движение в променливите, след което цялото движение на героя се занулява, към променливата “fall velocity” се добавя 1300 или ако стойността е отрицателна се заменя с 1200 и накрая новите стойности се използват за добавяне на импулс към героя.

Ако е включена опцията за силен скок проверката ще пропадне. Вместо това има други функции, които променят силата на първия скок.

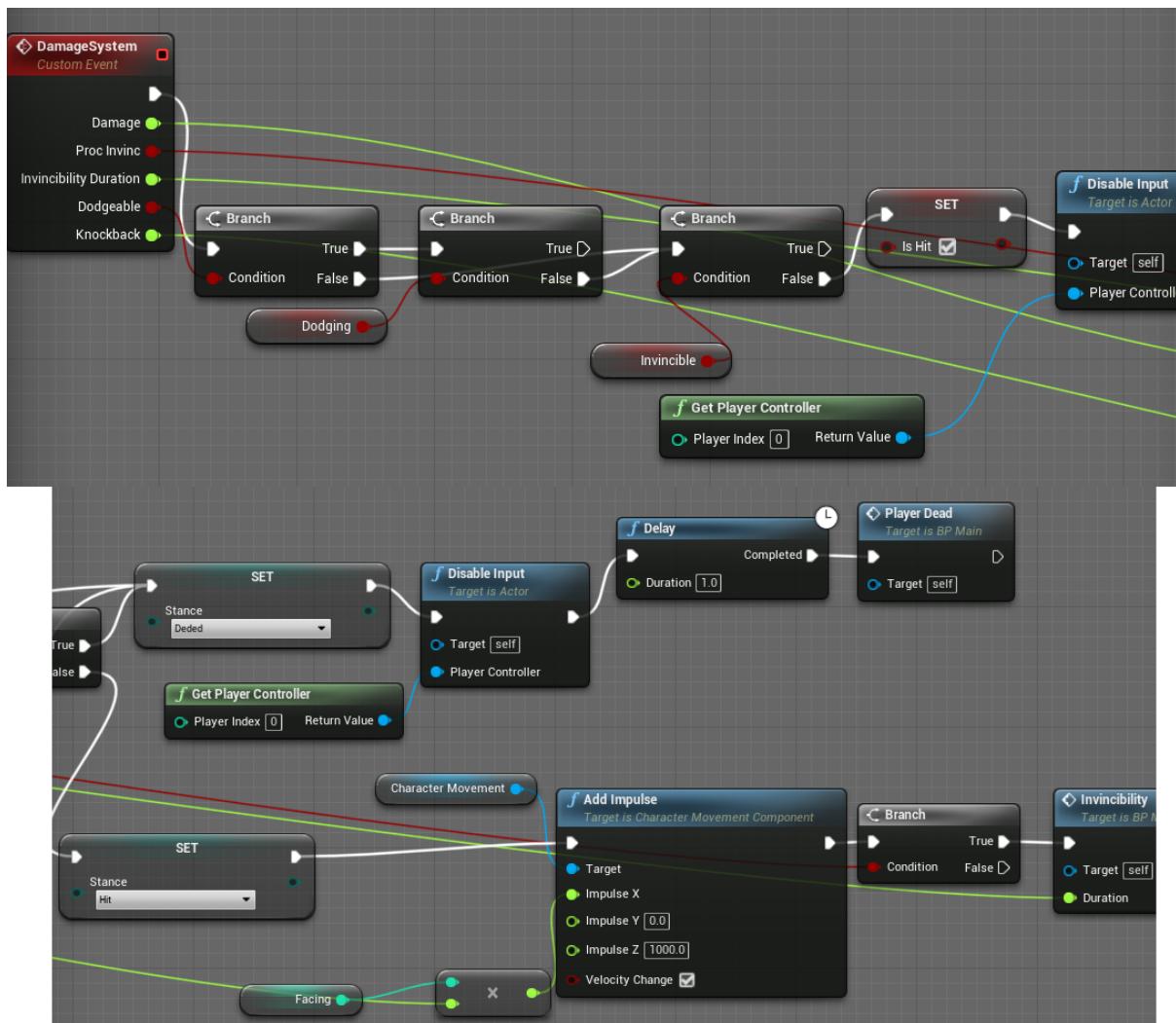
Система за поемане и възстановяване на щети

Втората основна система е тази за щети. Променливите, които тя използва са:

- “Health_current” от тип “float” със стойност по подразбиране “1.0”
- “Knockback” от тип “float” със стойност по подразбиране “-1000”
- “Invincible” от тип “boolean” със стойност по подразбиране “false”
- “BushUpgrade” от тип “boolean” със стойност по подразбиране “false”
- “BushUpgrade” от тип “boolean” със стойност по подразбиране “false”
- “Health_bar” от тип референция към обект от тип “HUD” със стойност по подразбиране “null”

Също към тази система има втори “Widget Blueprint” с име “HUD”, в който ще се съдържа интерфейс с визуализация на живота на героя. Чрез глобална функция се взима референция към героя и се взима стойността на “Health_current”.

“BP_main” има капсулна колизия по наследство, която се използва за засичане на сблъсъци с други обекти, но тъй като различните обекти нанасят щети по различен начин, то тук е единствено частта, която е отговорна за промяната на “Health_current” и за изхвърлянето на героя при удар. Тази функция има пет входни параметъра, които задават колко живот ще изгуби героя, колко силен да е тласъкът назад, дали ударът дава неуязвимост на героя, колко да е дълга тази неуязвимост и дали ударът може да се избегне.



Фиг. 18: Част от кода на функцията за поемане на щети

Другата функция тук е тази, която е отговорна за възстановяването на живот на героя и тя също е предвидена да се извика от външен обект и няма стойност по подразбиране. В тази функция се удвоава ефекта, ако героят има съответното подобреие.

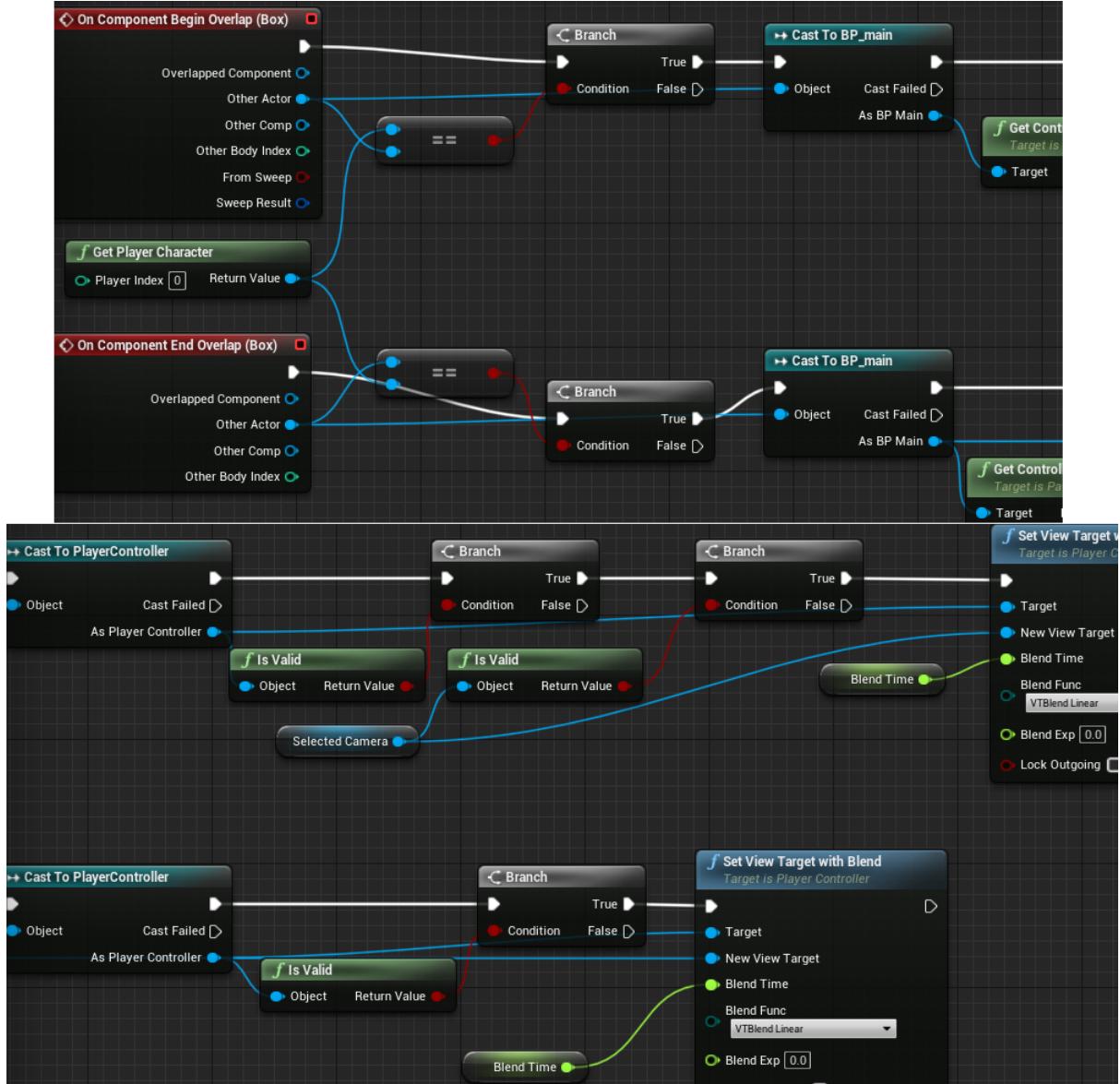
Външната част на тази система се състои от това друг blueprint да извика тези функции чрез глобална референция към “BP_main”, подавайки съответните параметри.

Система за динамична камера

Тази система се реализира чрез 2 външни blueprint-a. Единият е колизия, която се поставя в света и държи референция към другия, който съдържа статичната камера. Когато героят влезе в колизията, то колизията извика функция като и дава референция

към статичната камера и тя плавно сменя от камерата на героя към статичната камера.

При излизане от колизията се извиква друга функция, която прави обратното.

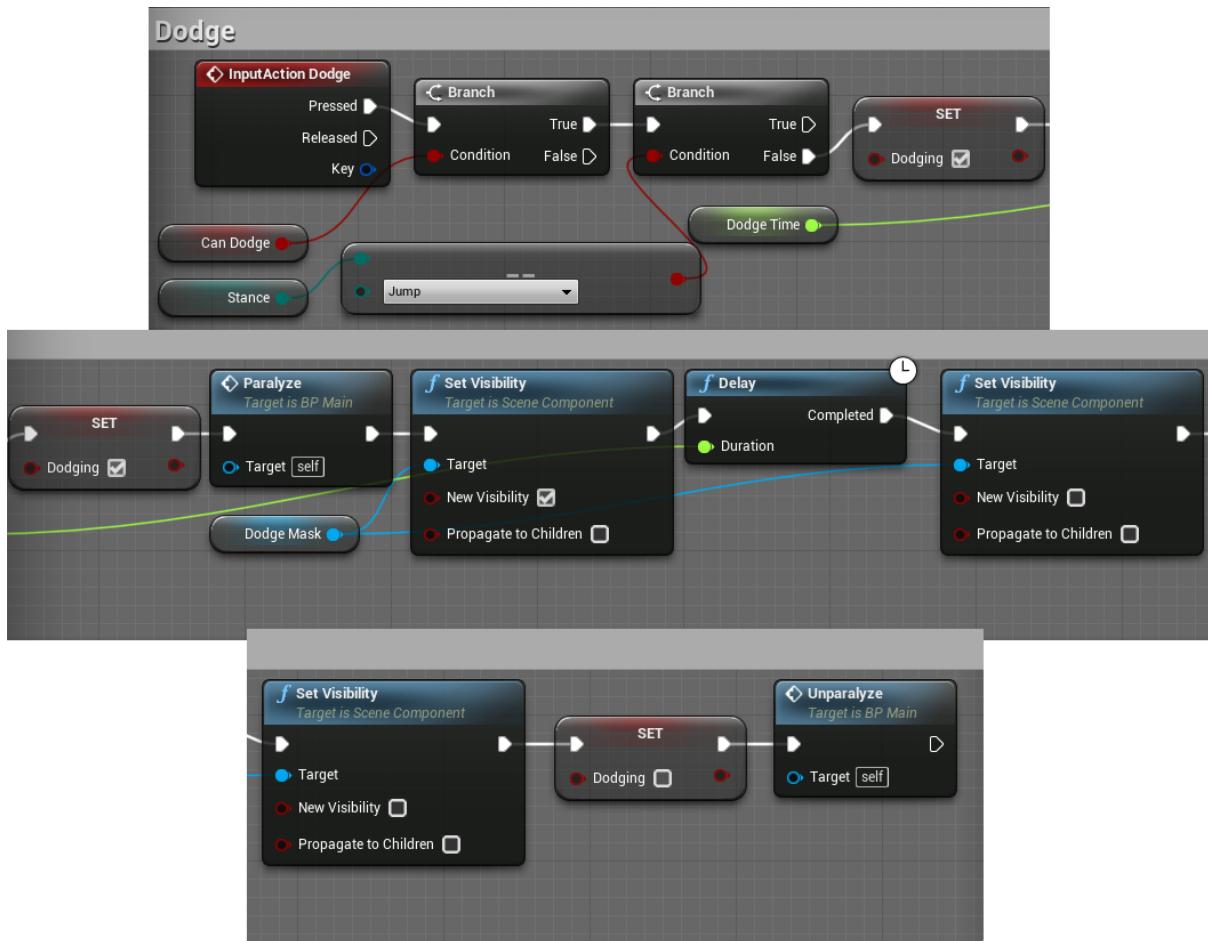


Фиг. 19: Част от кода на колизията на камерата

Система за избягване на атаки

Тази система се намира в “BP_main”. При натискане на “shift”, ако умението е отключено, се отнема контрол на героя от играла, задава се булиан и за 0.75 секунди героят избягва малки атаки, като жълъд хвърлен от катерица и визуално става леко по сив. След изтичане на времето отново се връща контрол на играла и героят се връща в

нормално състояние. По време на избягването по-големи атаки като атака от диво прасе не може да се избегне. Функцията е показана на фиг. 20.

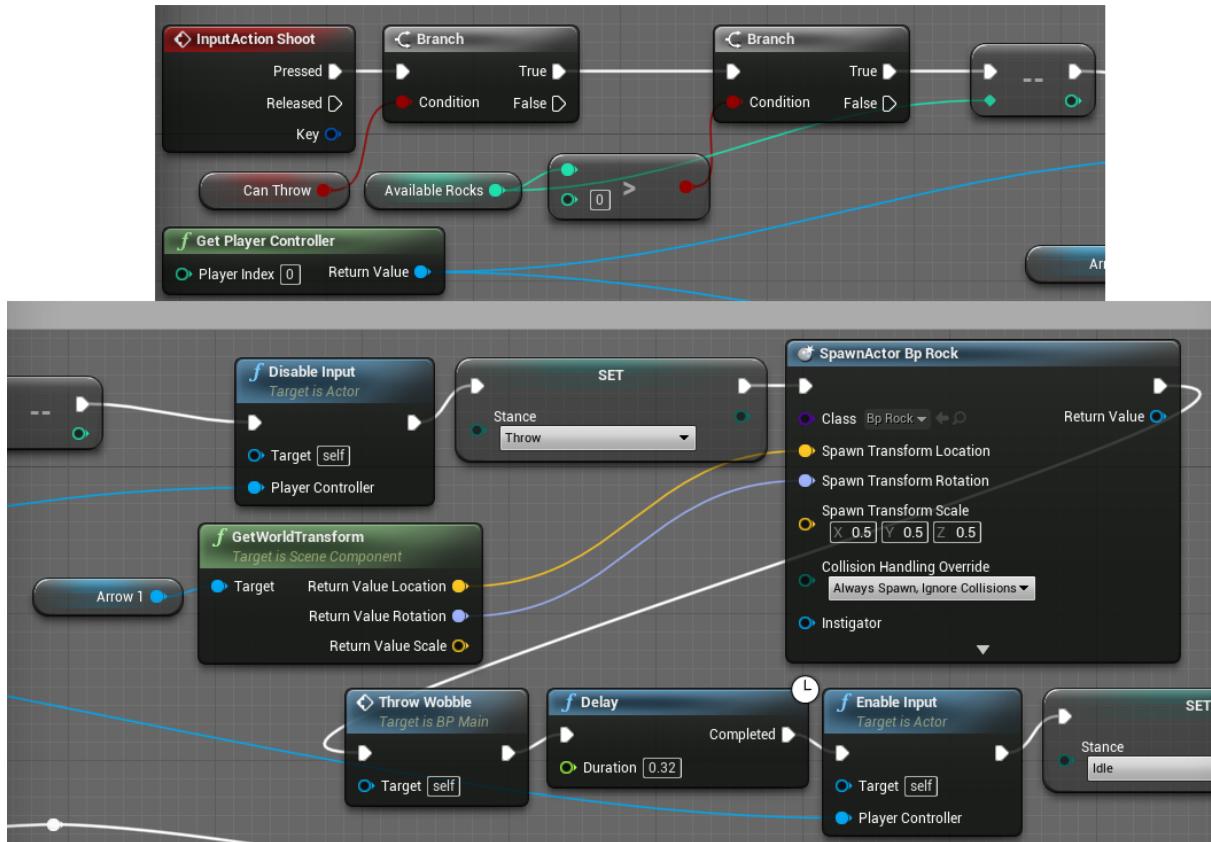


Фиг. 20: Функцията за избягване

Система за хвърляне на обекти

Тази система изисква още един обект - "Bp_rock". Този обект представлява камък, който използва компонента "InterpToMovement" за просто праволинейно движение. Камъкът се създава от функция на играла, която му дава правилна ориентация и посока в която да лети. Той също има вътрешна функция, която се активира при създаването на камъка и на всеки 20 милисекунди го завърта малко. Това създава анимация без да се налага да се променя изображението на обекта. Ако камъкът не се удари в нищо за 5 секунди или се удари в терен, то той се самоунищожава. Катериците, които се гонят чрез тези камъни имат своя функция за засичане на удар с камък. Има булиан, който

роверява дали героят е отключил умението да хвърля и дали има налични камъни. При отключване на умението, се изпълнява функция, която добавя визуален индикатор за броя камъни към “HUD”. Последният обект от тази система е купчина от камъни, от която играчът може безкрайно да се сдобива с камъни. Когато играчът влезе в колизията на купчината броя налични камъни, който се държи в променливата “AvailableRocks” става 5.



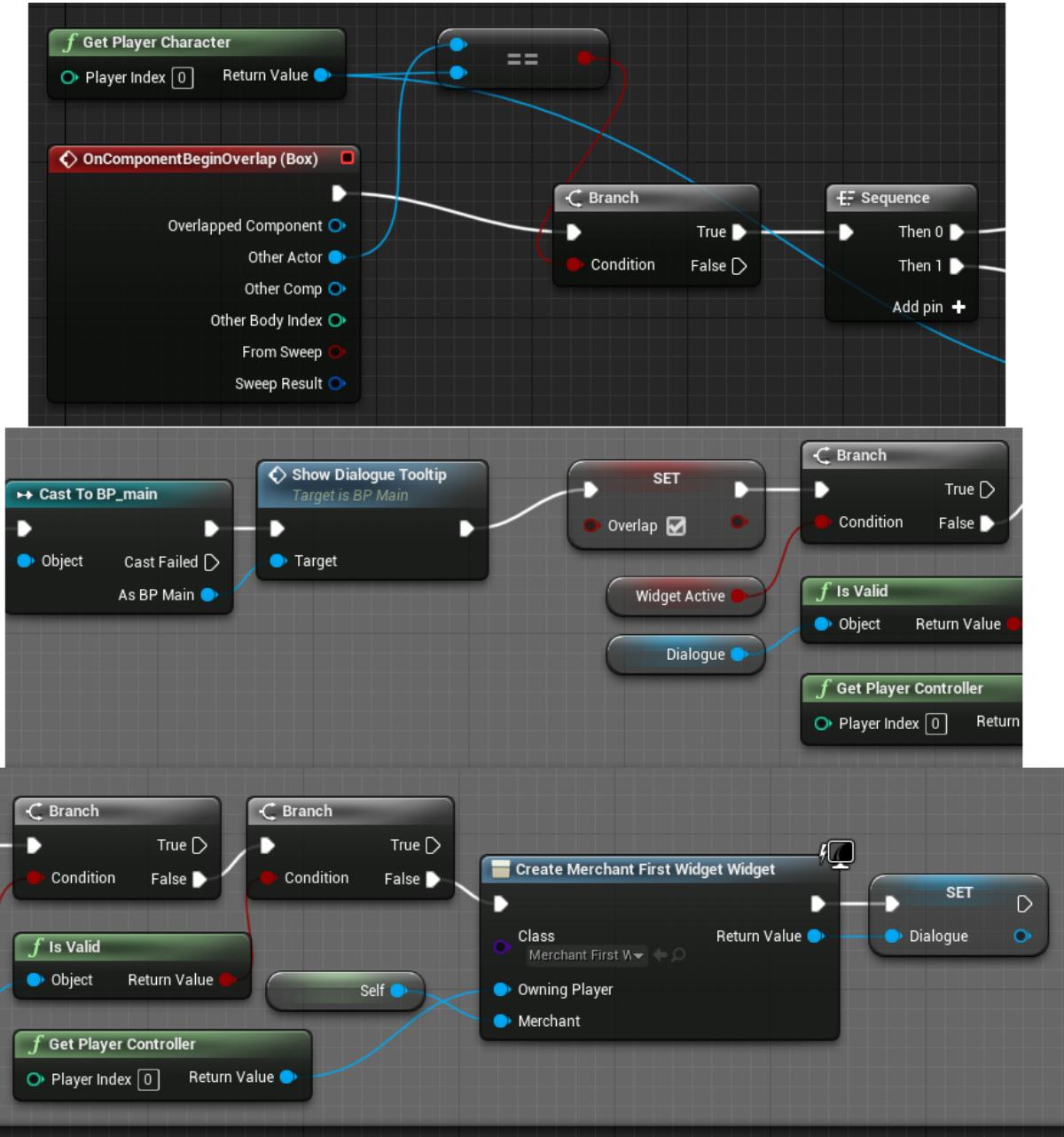
Фиг. 21: Функцията за хвърляне на камък в “BP_main”

Система за диалог

Системата за диалог е доста голяма и за нейната работа са нужни функции в “BP_main” в blueprint-а на нивата, които съдържат диалог и още два blueprint-а за всеки отделен разговор или събитие, плюс странични обекти, които се влияят от разговора като например невидими стени.

Първата част е в “BP_main”, където се засича натискането на клавиша “F” и при него за кратко променя булевата променлива “AttemptInteract” на “true”, и след 200

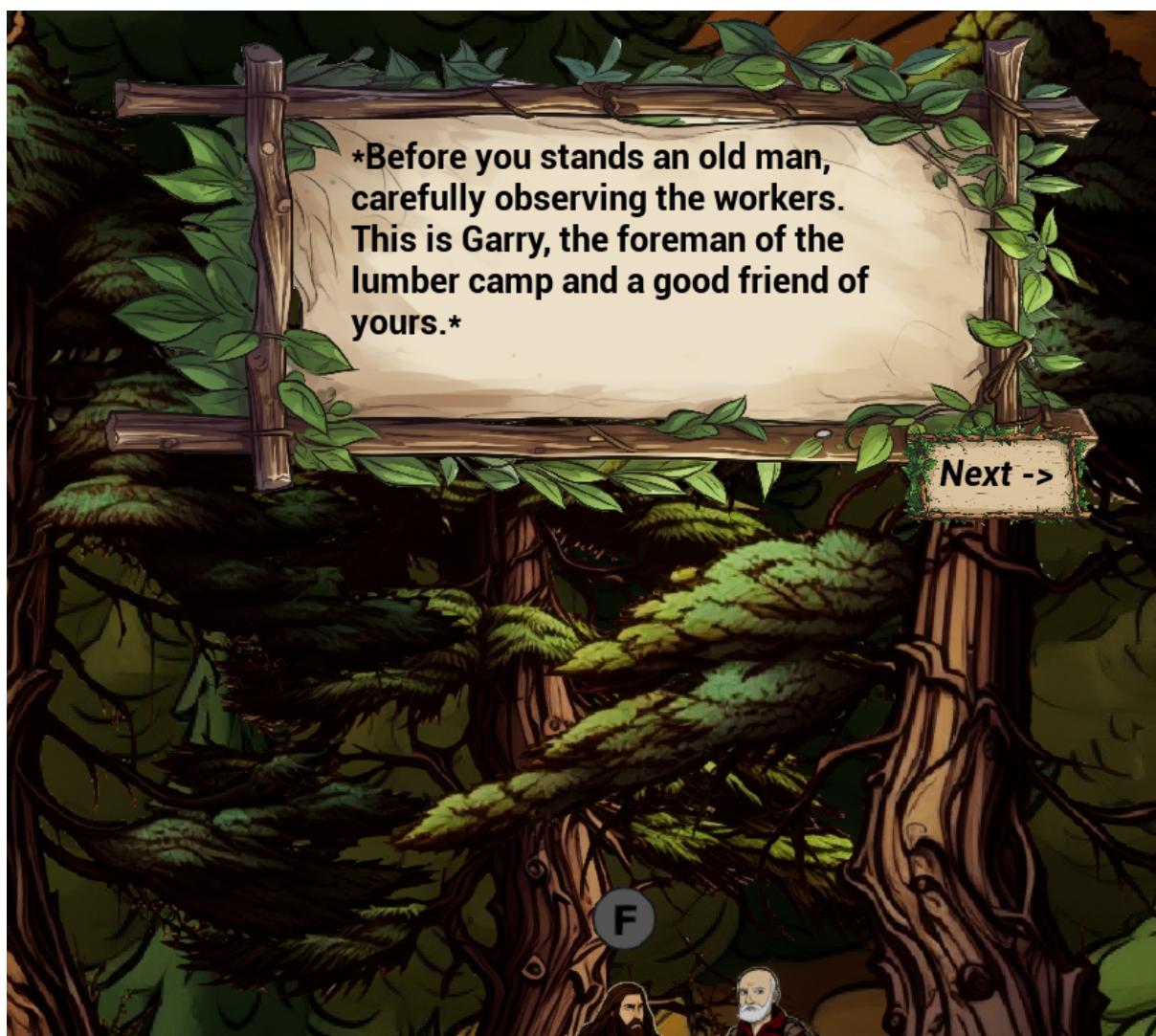
милисекунди отново на “false”. Тази булева променлива се гледа от другите обекти. Тези обекти си имат 2 части. Първата е blueprint, който се поставя в света и има колизия и графичен елемент. При колизия с играча, започва да проверява през 100 милисекунди състоянието на “AttemptInteract” променливата. Също така при първата колизия се приготвя обект от принадлежащия “Widget Blueprint”.



Фиг. 22: Кода на колизията, който създава обекта на интерфейса

При засичане на опит за начало на разговора колизията парализира героя и показва мишката на экрана, след което изкарва интерфейса за разговор. Този интерфейс

съдържа кутия с текста на дискусията и копче за продължаване на диалог. Има и диалог, който се променя в зависимост от това, което героят казва и в този случай се скрива копчето за продължаване и се появяват копчета със съответните избори. Целият текст се съдържа в масив от текстови променливи и се изваждат от там с цикъл до край на масива. Ефектите от диалога могат да се вържат както за определена реплика, така и за края на диалога. Подобренията, които се дават на героя чрез диалог се прилагат чрез глобални функции. Също така има подсказка за играта, когато е вътре в колизия на диалог, която показва кръг с буквата "F" над главата на героя.



Фиг. 23: Диалоговият прозорец по време на диалог

Последната част от тази система се случва в така нареченият "Level Blueprint", където на диалога се дават референции към обектите, които ще променя.

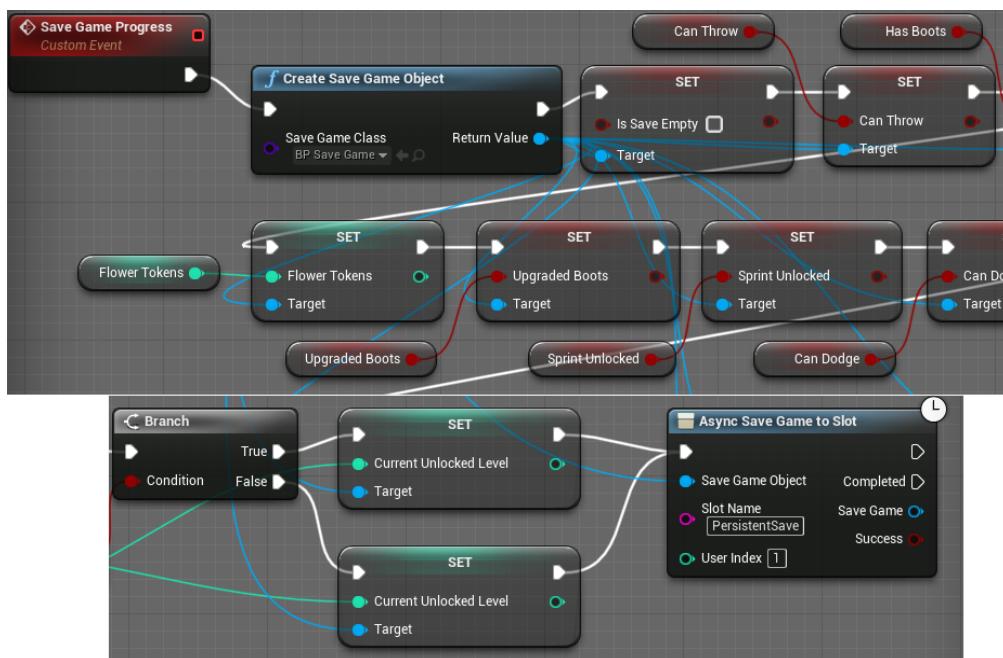
Система за запазване на прогрес

Тази система работи чрез наследяване на класа “Save Game” на Unreal Engine. В новия blueprint са дефинирани следните променливи от “BP_main”

- “CanThrow” от тип “boolean”
- “HasBoots” от тип “boolean”
- “FlowerTokens” от тип “integer”
- “UpgradedBoots” от тип “boolean”
- “SprintUnlocked” от тип “boolean”
- “CanDodge” от тип “boolean”
- “BushUpgrade” от тип “boolean”
- “CurrentUnlockedLevel” от тип “integer”

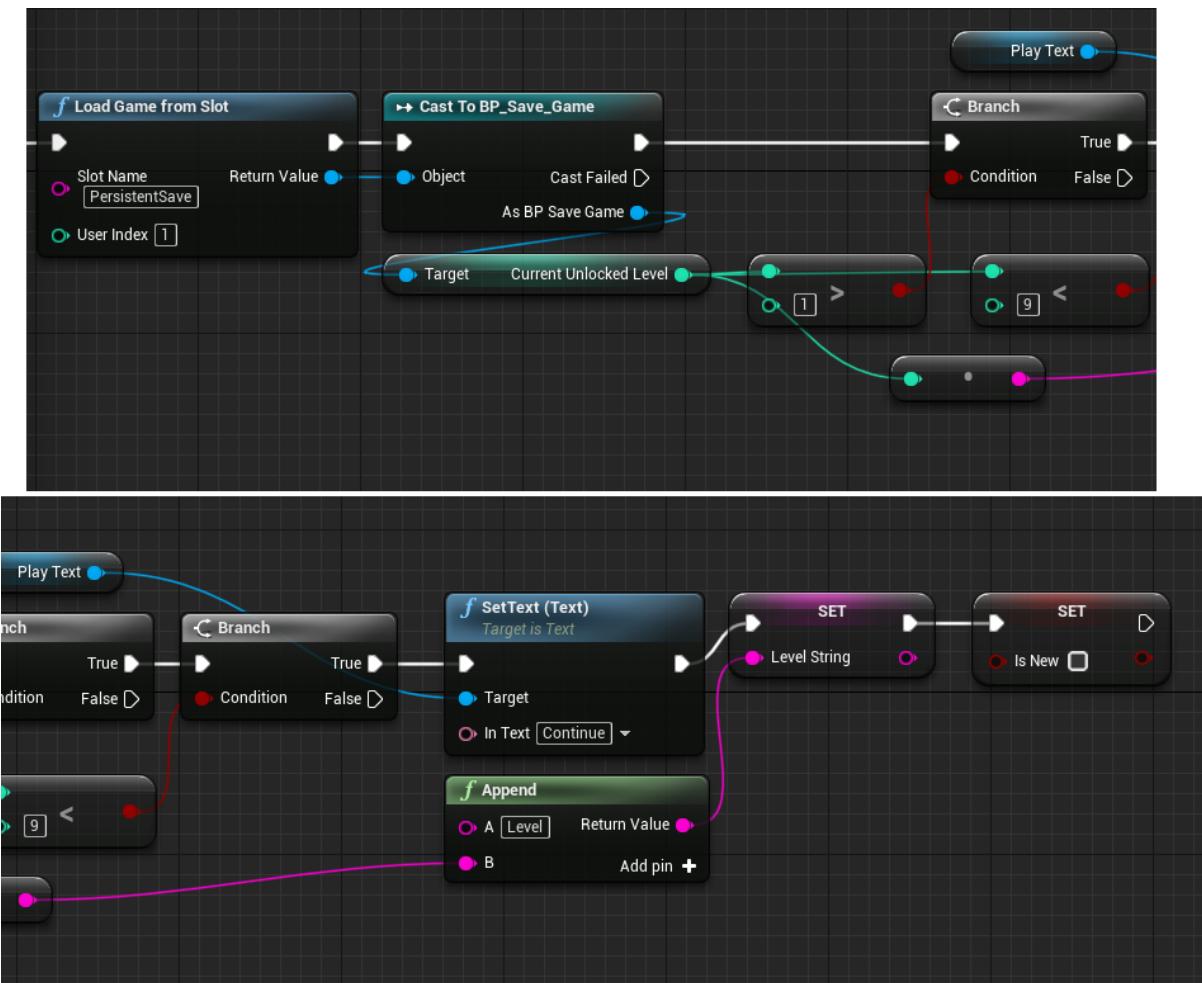
В допълнение към тях има и още една булева променлива “IsEmpty”, която се използва за проверки при записване и зареждане на прогрес.

Записването на прогрес се извършва вътре в “BP_main” чрез функцията “Create Save Game Object”, и се избира класа, в който са деклариирани променливите като шаблон. След това чрез “Set” функциите на всяка променлива ги задаваме да са равни като тези на текущия герой. След записа се променя “IsEmpty” на “false” и се извиква функцията “Async Save Game To Slot”, като за име се дава “PersistentSave”, а за номер на слот “1”. Тази функция за запис се извиква от знака, който служи за край на ниво, чрез референция към “BP_main”.



Фиг. 24: Началото и края на функцията за запазване

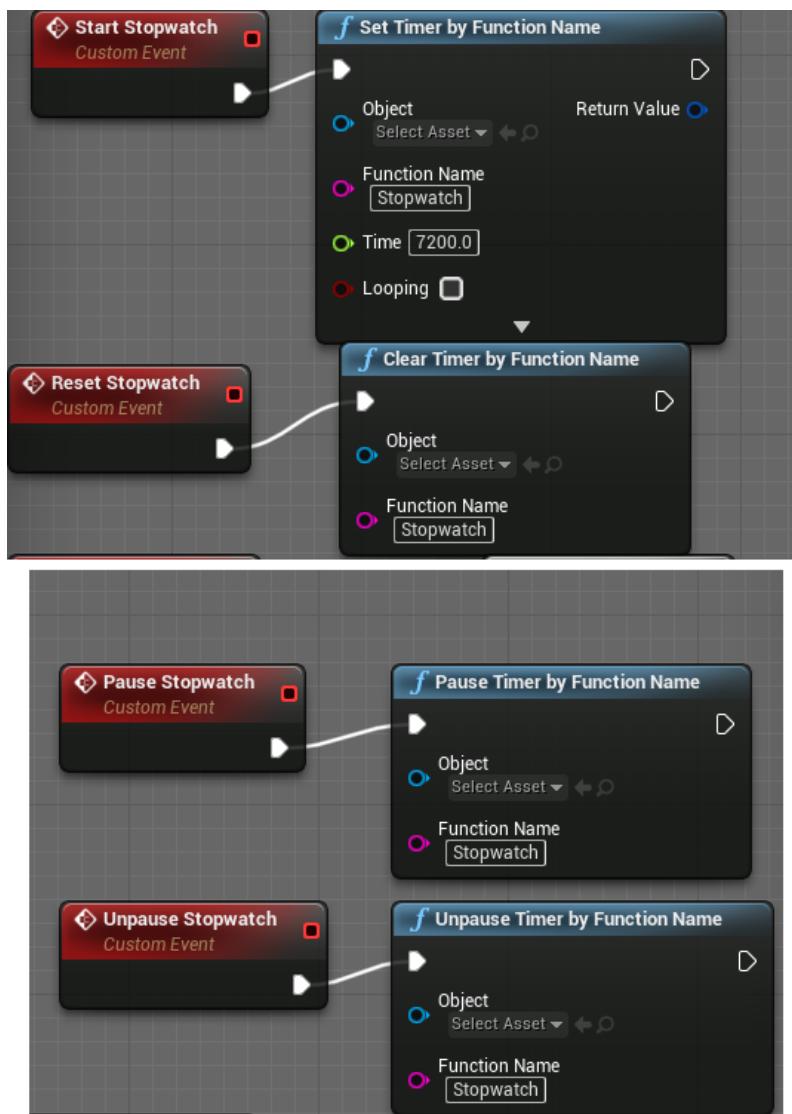
Другата функционалност е зареждане на съответните данни. Това се извършва отново в “BP_main” и в някой менюта, чрез функцията “Load Game From Slot” и за име се дава “PersistentSave”, а за номер на слот “1”. След това изхода се превръща в обект от по-рано създаденият “Bp_Save_Game” клас. Когато обекта е създаден се проверява стойността на “IsEmpty”. Ако е “true” обекта се изхвърля и се използват стойности по подразбиране, а ако е “false”, то от обекта вече може да се изважда информация чрез “Get” функции. В “BP_main” тази функция е една от първите която се изпълняват при начало на игра, тъй като при смяна на ниво всички обекти се унищожават и на новото ниво героят се създава с настройки по подразбиране. Също се използва и в основното меню, за да определи дали на първото копче да пише “New Game” или “Continue” и в менюто за избор на ниво, за да определи кои нива са отключени.



Фиг. 25: Определяне на това дали играта е нова в основното меню

Система за запазване на рекордно време

Тази система се състои от две части. Първата част е отговорна за запис и зареждане на резултатите и тя работи подобно на системата за запис на прогрес. Втората част е хронометър, който засича времето от начало на нивото до докосване на знака за край. Този хронометър съществува като група от функции в “BP_main”. За щастие в Unreal Engine вече има функционалност за следене на време от екзекуцията на определена функция. Първо се дефинира една празна функция “Stopwatch”. След това функцията “Start Stopwatch” извиква функцията на Unreal Engine “Set Timer By Function Name” с параметър името на празната функция “Stopwatch”. Същото се прави със съответните функции на Unreal Engine, за да се реализира способност за пауза, рестарт и вземане на резултат от хронометъра.



Фиг. 26: Основни контролни функции на хронометъра

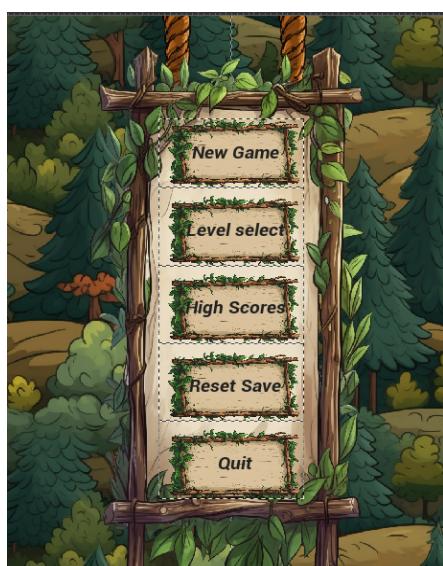
При пауза на нивото чрез “escape”, хронометърът също спира. Съответно продължава, когато менюто се махне. При приключване на ниво част от процедурата вика функцията за спиране на хронометъра и за форматиране на неговият резултат от милисекунди в по-четимо за играча време по формат ”минути:секунди:хилядни” и този резултат се предава на частта за запис.

Имплементация на менютата

Всички менюта използват “Widget” класовете на Unreal Engine. Когато един blueprint е създаден като “widget”, той има два редактора. Първият си е стандартният редактор където се пише кода, а втория е визуален редактор за подреждане на елементите на менюто. Всички менюта следват един формат. Той има следните елементи:

- Фон, за който се използва изображение, което се повтаря незабележимо от всички страни
- Изображение на хартиена таблица, което служи за рамка и фон, на който да се поставят елементи
- Бутони с различни функции

Има разлики между броя бутони и табели в различните менюта. Например в менюто за избор на ниво има 2 табели, за да има място за повече бутони. Отварянето и затварянето на тези менюта става главно чрез референции в обекта на главния герой. В основното меню също има опция за изчистване на запазения прогрес и рекорди. Менютата също така се опитват да променят размера си динамично, за да се адаптират към резолюцията.



Фиг. 27: Основното меню във визуалния редактор

Непреценени разработки

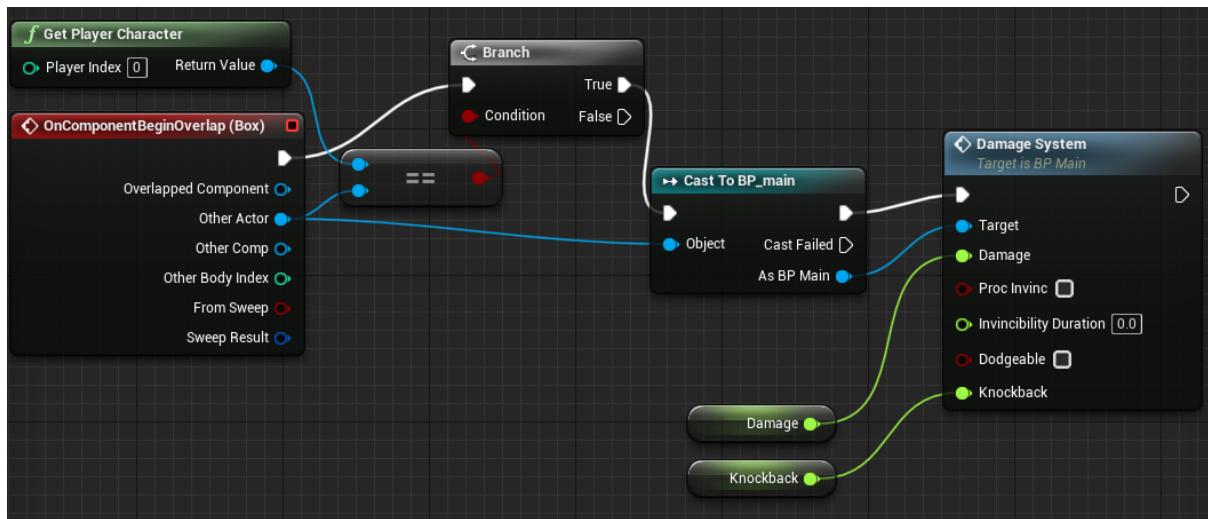
По време на разработката на архитектурата на приложението бяха останали елементи на играта, които бяха предвидени, но не точно оточнени и такива които изобщо не бяха планирани, но се появиха нужда от тях по време на разработка.

Неоточнени разработки

Първата група от тези разработки съдържа вражеските същества и капани

Тръни

Това е първият капан, с който играчът се среща. Състай се от графичен елемент и колизия. При докосване на героя до колизията се извиква чрез референция “Damage System” функцията на главния герой и той поема 15% от живота си като щети и се изхвърля назад. Този капан не предоставя неуязвимост и когато са поставени в група могат да са доста опасни, тъй като отскокът от единия може да прати героя в следващия.

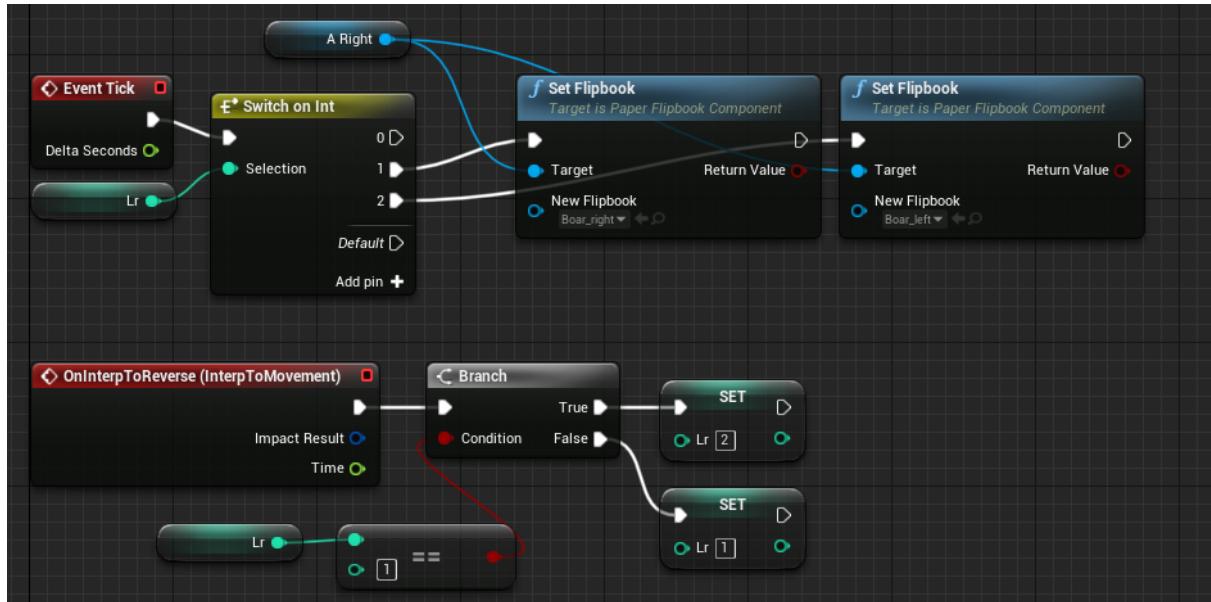


Фиг. 28: Функцията която се изпълнява при колизия

Диво прасе

Дивото прасе използва компонента “InterpToMovement”, за да се движи. Разстоянието, което патролира и скоростта, с която се движи е направена да може да се настройва индивидуално за всяко прасе в редактора за нива. Също има опция за забавено начало на движението с произволно време, за да се избегне 2 прасета да се движат в синхрон.

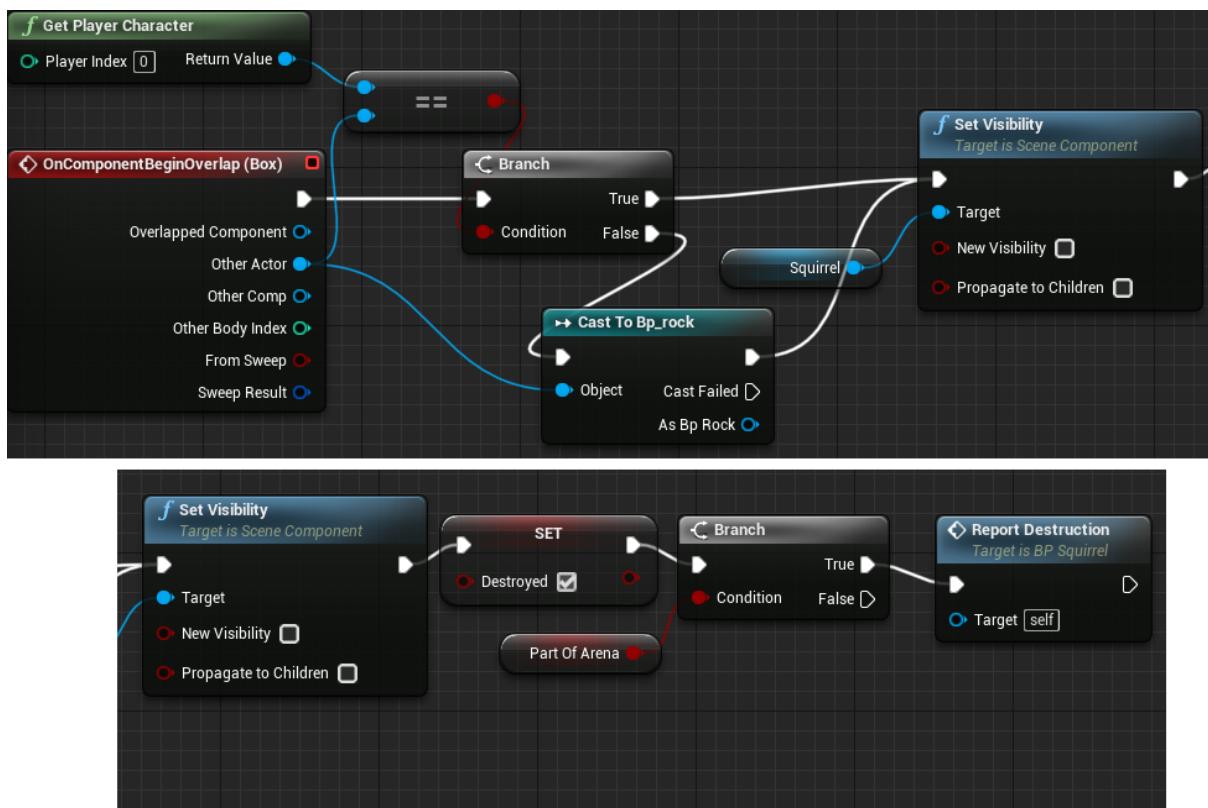
Също има и функция, която следи прасето да гледа в посоката, към която се движи. Прасето има колизии отпред и отзад, който удрят играча чрез “Damage System” за една четвърт от живота му. Ударът също дава неуязвимост за половин секунда и спира движението на прасето за 1 секунда. Прасето също има колизия на гърба си, която работи като терен и дава възможност на играча да язди прасето и да се възползва от неговата инерция при скок.



Фиг. 29: Управление на ориентацията на прасето

Катерици

Катериците не се движат и не удрят при допир, а вместо това хвърлят жълъди през интервал, който може да се настройва индивидуално. Също за всяка катерица може да се зададе забавяне на началото на стрелба и дали да са върху пън или не(ако са поставени на дърветата от фона). Хвърлянето на жълъди представлява създаване на обекти от тип “Bp_acorn” и задаване на посока на летене. Има два вида катерици - тези които хвърлят хоризонтално и тези, които пускат жълъди надолу. Жълъдите на катериците могат да се избегнат чрез системата за избягване на главния герой. За да се отърве играчът от една катерица трябва или да влезе в нейната колизия или да я улучи с хвърлен камък, което извиква функцията за унищожение на обекта. Също както при камъните хвърлени от героя, жълъдите имат функция която ги върти, за да създаде анимация. Жълъдите не дават неуязвимост при удар.



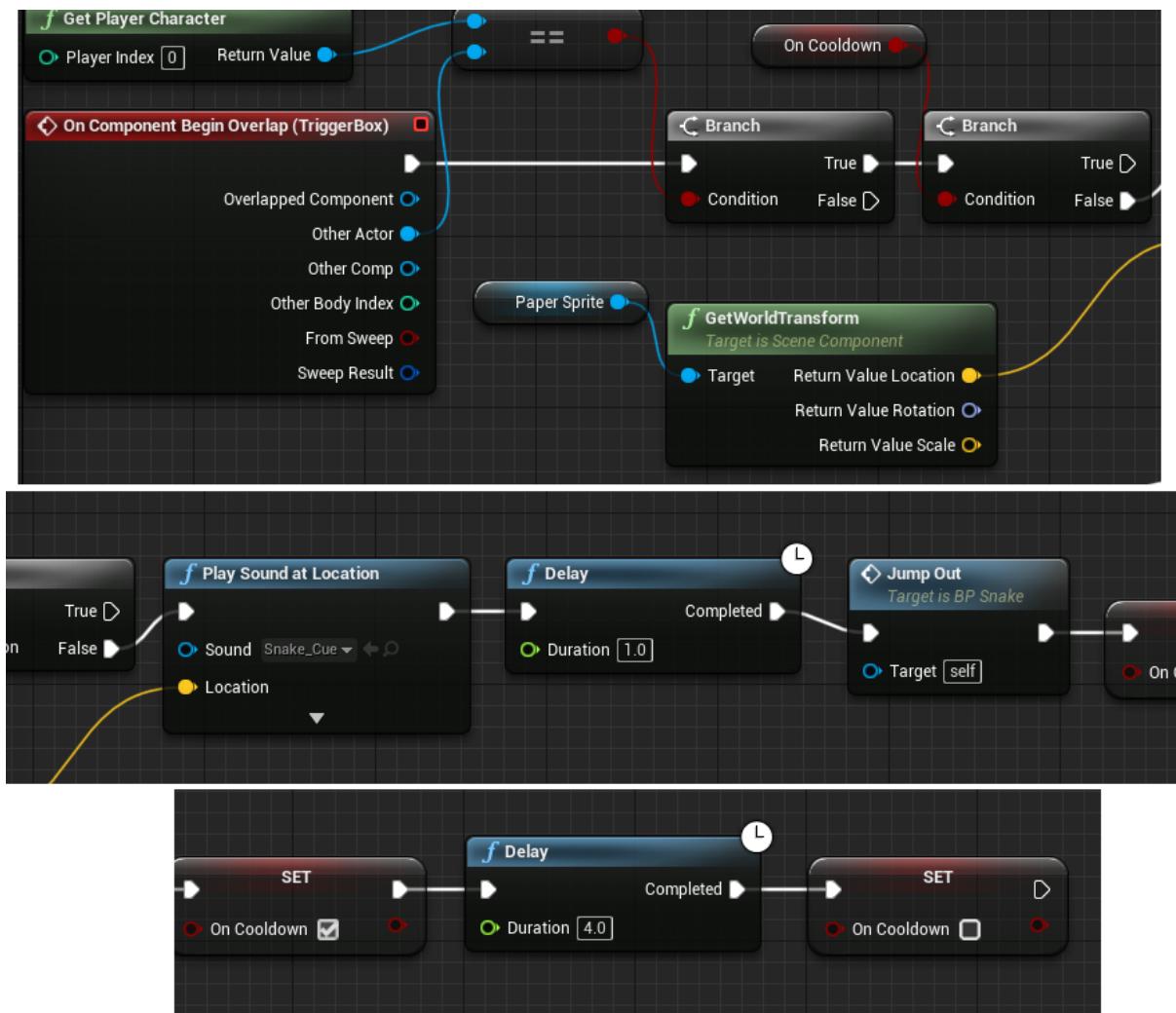
Фиг. 30: Функция за унищожаване на катерица

Летящи тръни

Летящите тръни са нещо средно между терен и капан. Както името подсказва те се поставят във въздуха и имат колизия за удар както отгоре, така и отдолу. Разликата между двете колизии е, че горната колизия работи както нормалните тръни, а долната колизия нанася щети, но вместо да хвърля героя назад, тръните просто премахват всяка инерция от героя, което го кара да падне на земята. Тези тръни се използват за ограничаване на използването на големият скок на героя на определени места.

Змия

Змията е нещо средно между капан и създание. Тя има голяма и малка колизия. При навлизане на играла в голямата колизия змията издава съскащ звук и след 1 секунда изскочи над земята заедно с малката колизия. Ако играчът докосне малката колизия, то той поема щети чрез “Damage System”. След изскочане змията се скрива отново и се задава период от 4 секунди, в които не може да бъде активирана пак.



Фиг. 31: Функция за изскачане на змията

Плодов храст

Този храст се използва, за повишаване на здравето на героя. Състой се от графичен елемент и колизия. Когато героят влезе в колизията, храстът извършва проверка дали здравето на героя е под 90% и ако това е така, той възстановява 15% или 25%(с подобрене) здраве на героя. След това графичният елемент на храста се променя за да изглежда обран и той става неактивен.

Втората група неоточнени разработки са различни видове терен. Базов терен

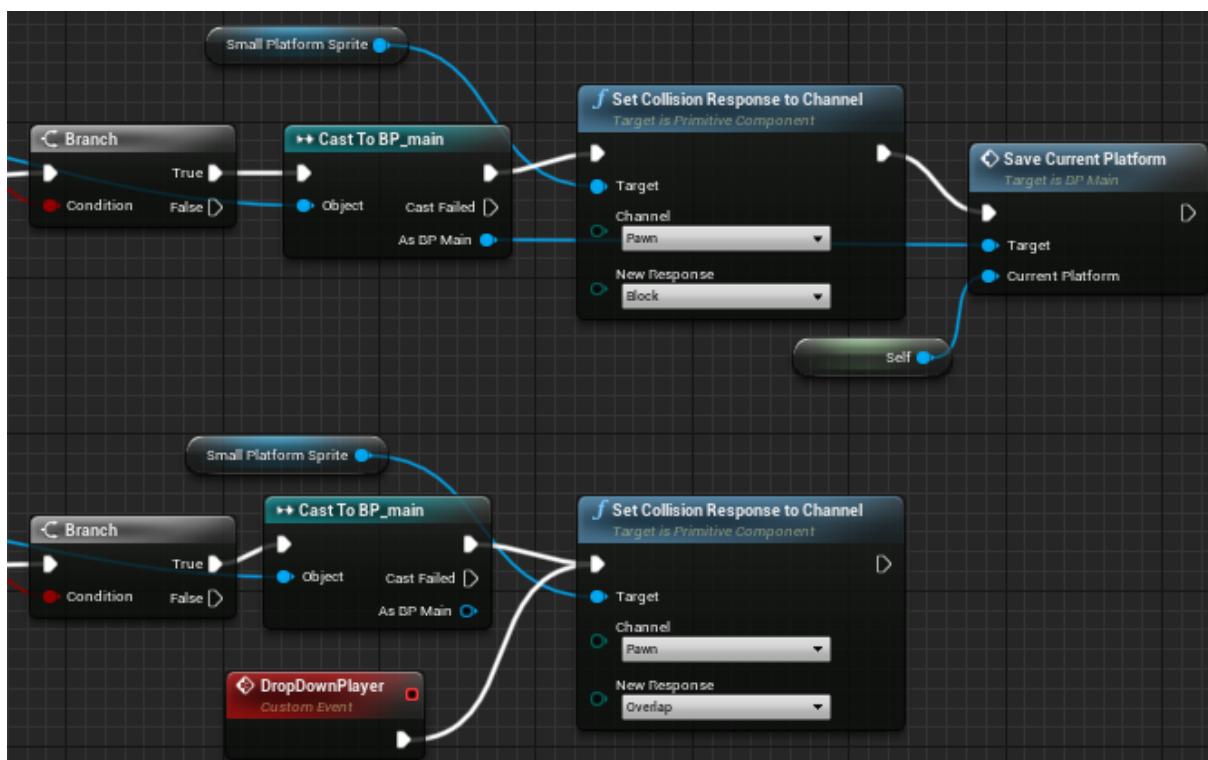
Както името подсказва това е теренът, който сформира основата на нивото. При него няма нищо особено. Той е просто графичен елемент с колизия, която не пропуска нищо да мине през нея.

Невидима стена

Невидимата стена е вертикална колизия без графичен компонент и съдържа функции за включване и изключване. Най-често се използва за поставяне на граници в края на нивата и за ограничител на част от картата, който може да се махне от диалог.

Летяща платформа

Тази платформа се използва за вертикално разширяване на нивото. На тях могат да се поставят вражески същества и капани. Те се състоят от графичен елемент, теренна колизия покриваща графичния елемент и колизия, която е над графичния елемент. По подразбиране теренната колизия позволява на главният герой да минава през нея. Ако героят стъпи отгоре на графичният елемент, горната част на капсулата на героя докосва летящата колизия, която извиква функция, която прави долната теренна колизия солидна и вече не може да се минава през нея. Това позволява на героя да стои върху платформата. Ако капсулата на героя спре да докосва горната колизия, то се изпълнява друга функция, която отново прави теренната колизия пропусклива. Героят може да слезе през всяко време от платформа чрез натискане на "S". Това чрез запазена референция запазена от по-рано извиква функцията на платформата за пропускливост.



Фиг. 32: Функции за активация и деактивация на платформата

Движеща се платформа

Същата функционалност като стандартната платформа, но има “InterpToMovement” компонент. Може да се движи от точка до точка. Точките, скоростта и време за забавяне преди старт на движение могат да се настройват индивидуално за всяка платформа в редактора на новото.

Последната група неуточнени разработки са приятелските хора.

Дискусииите с хора обикновено отключват невидима стена или дават възможност за подобрение. Те също служат за разясняване на света на играта.

Непредвидени разработки

Обучаващи невидими колизии

Обучаващите невидими колизии се използват, за да обяснят на играча как нова механика в играта работи. Те се състоят от колизия, подобна на тази за хората и съответен “Widget” обект. За разлика от колизията на хората тази колизия игнорира състоянието на “AttemptInteract” и директно започва монолог.

Други

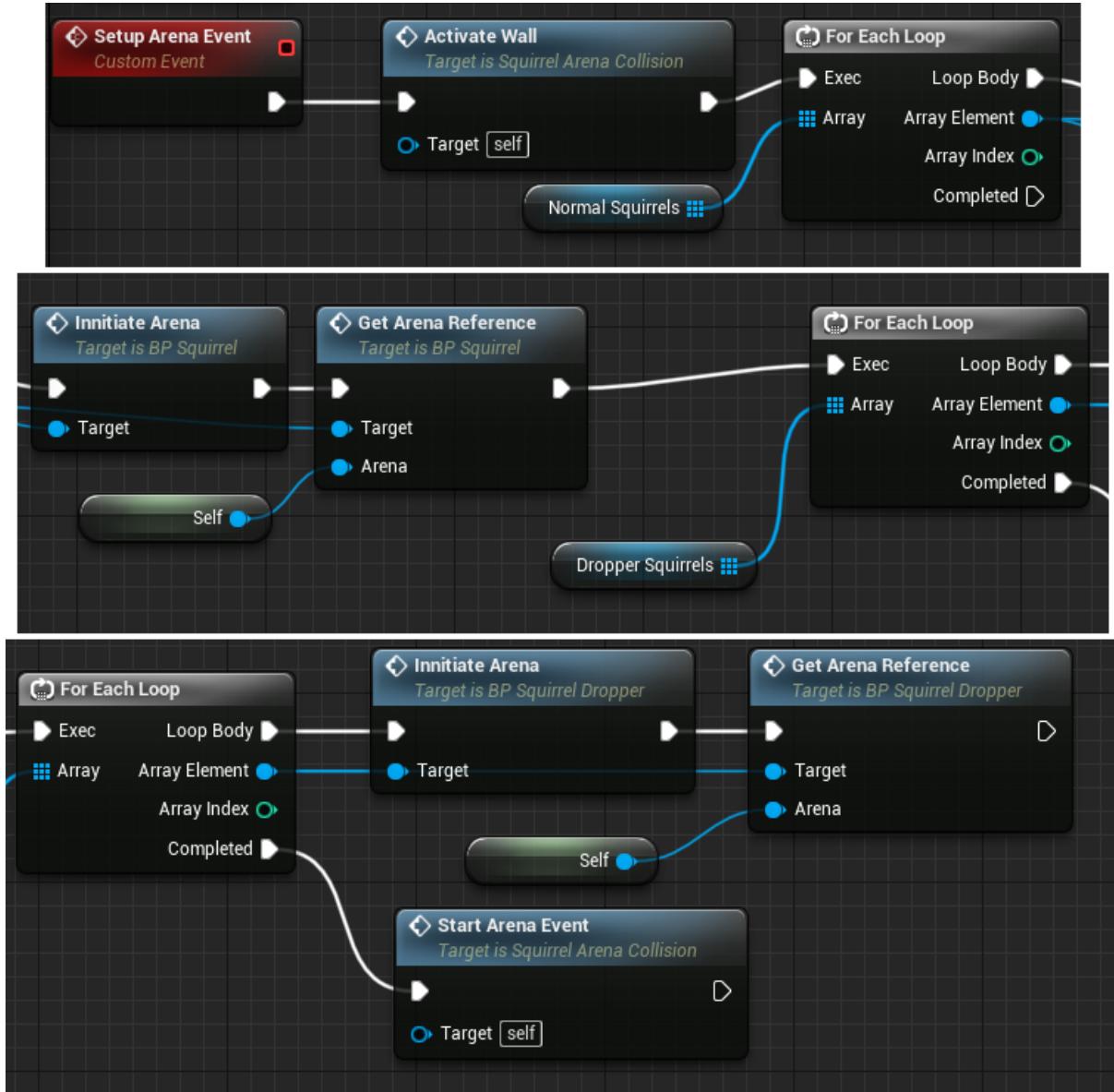
В групата на другите се съдържа експозицията в началото на играта, цвете, което може да се размени за подобрение, колизията, която започва последната битка на играта, самата битка и анимации

Последната битка

За финал на играта героят трябва да спечели битка срещу група катерици. Логиката за битката се съдържа в колизията за начало и за битката също има специален интерфейс, който показва броя победени катерици и таймер, при изтичането на който героят умира.

При начало на битката се активира интерфейсът като чрез референция се извиква функция в “HUD”. Таймерът също започва да отброява надолу и групи от катерици започват да се появяват. Катериците имат достъп до допълнителни функции, ако булиана “PartOfArena” е със стойност “true”. Може да се настрой времето, което ще им трябва, за да се включат в битката и имат референция към колизията за старт на която

докладват, ако умрат. При успешно изгонване на всички катерици таймерът спира, излиза съобщение за победа и на мястото на колизията за начало се появява синьо цвете, при контакт с което излиза съобщение, че това е края на демото и нивото завършва.

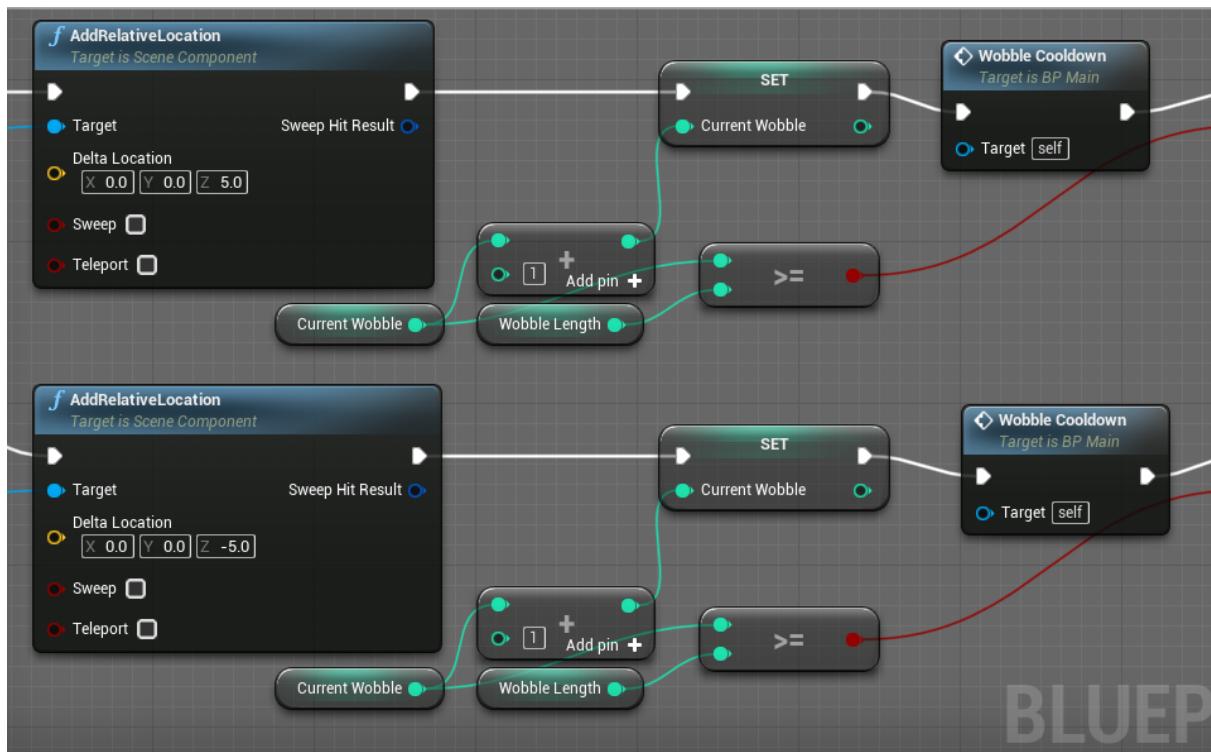


Фиг. 33: Функция, която подготвя битката и катериците

Анимации

Голям проблем с използването на графика ексклузивно направена чрез дифузен модел, е че тези модели не работят по начин, който е съвместим със създаването на анимации. Midjourney има функция наречена “remix”, която се опитва да постигне анимация, чрез редакция на текста на предишна генерация, за да се посочи действието, което трябва да

се случи. Това създава горе-долу приемливи резултати при създаването на анимации на човешко лице, което да си промени изражението или да оства с всяка следваща генерация, но дори и тези анимации имат очевидни разлики между всеки кадър. Анимациите нужни за игра за момента не са възможни със сегашните способности на модела. По тази причина трябваше да се измисли друг начин за анимиране на най-основните движения в играта. Това се прави с циклични ротации или локални измествания на графичния елемент на обект. Анимациите направени по този начин в играта са движението на главния герой, хвърлянето на камъни, хвърлянето на жълъд от катерица и въртенето на камъка и жилида.



Фиг. 34: Част от кода за анимиране на ходенето на героя

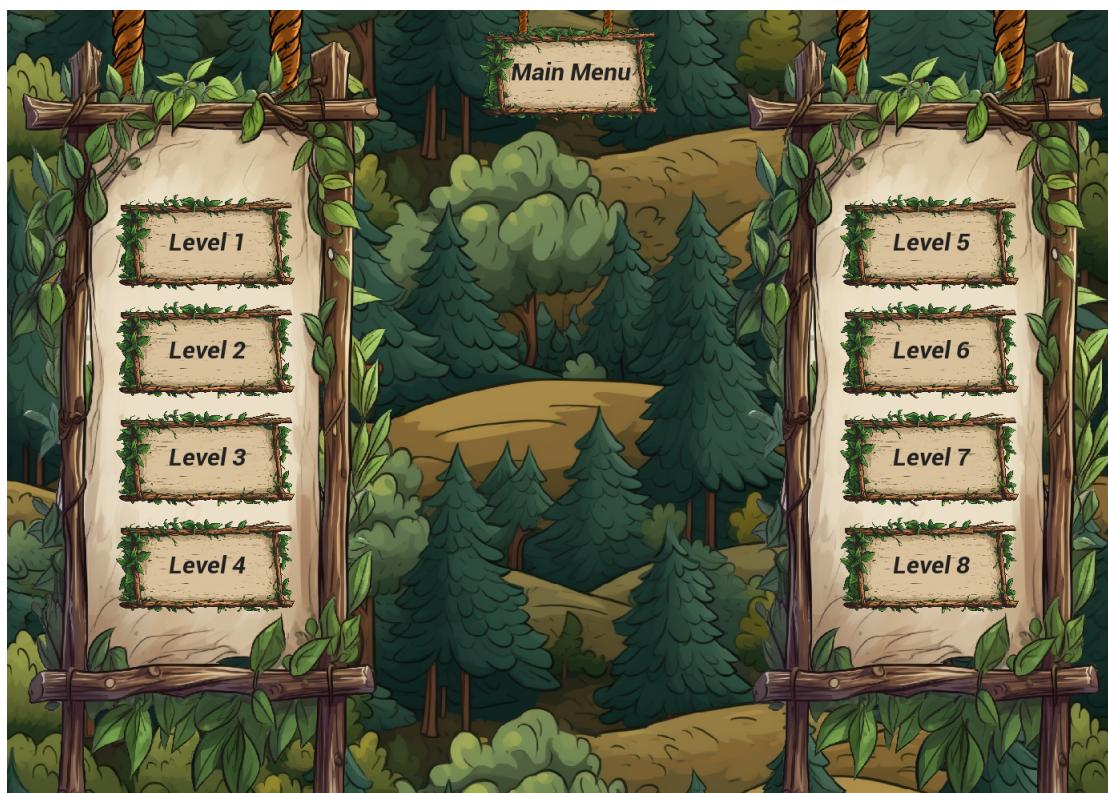
Ръководство на играта

“*The Secret Almanac*” е 2D sidescroller игра, в която дървесекач се озовава в непозната част от гората на път за вкъщи. Там среща мистериозен индивид, който му помага да се справи с препятствията, които стоят на пътя му и му предлага различни странни артефакти, които правят невъзможното възможно.

Начало на играта

При стартиране на играта първото нещо на екрана е основното меню на играта. Тук има следните копчета:

- “New Game” за начало на играта
- “Level Select” за отваряне на менюто за избор на ниво, където може да се стартира определено ниво след като то бъде отключено
- “High Scores” за отваряне на изглед с рекордните времена за завършване на всяко ниво
- “Reset Save” за изчистване на прогреса на играта и рекордните времена
- “Quit” за затваряне на играта



Фиг. 35: Меню за избор на ниво

Първо ниво

След натискане на “New Game” в основното меню, се зарежда първото ниво на играта. В това ниво няма вражески обекти и целта му е да запознае играча с основното движение на героя и да въведе диалоговата система на играта. Чрез нея се прави експозиция на света на играта. След като завършат диалозите героят може да продължи надясно до достигане на знак, който означава края на нивото. Също така по всяко време може да се натисне “Escape”, за да се постави играта на пауза и да се изкара меню за пауза. При достигане на края на нивото се отваря менюто за край на ниво. От него може да се продължи към следващото ниво с копчето “Next Level”, да се върне в основното меню с копчето “Main Menu” и копче “Quit” за затваряне на играта.



Фиг. 36: Меню за край на ниво

Второ ниво

Във второто ниво за първи път се срещат опасности. Първо по пътя героят намира тръни. За да се премине през тях трябва те да се прескочат, като се внимава да не падне героя върху тях. Ако героят падне върху тръните, той поема щети и се изхвърля назад. Второто препятствие в нивото са диви прасета. Те патрулират определен район и могат да удрият играча с главата и задните си крака. Гърбът на дивите прасета обаче е безопасен и на него може да се стъпва. При успешно минаване на тези препятствия се достига до края на нивото.



Фиг. 37: Героят язди диво прасе

Трето ниво

В третото ниво за първи път се среща мистериозен човек, който обяснява на героя за случващото се и му дава чифт нови обувки. След преминаване на тръни и група прасета, героят ги обува и разкрива, че вече може да скача по два пъти. Това умение ще

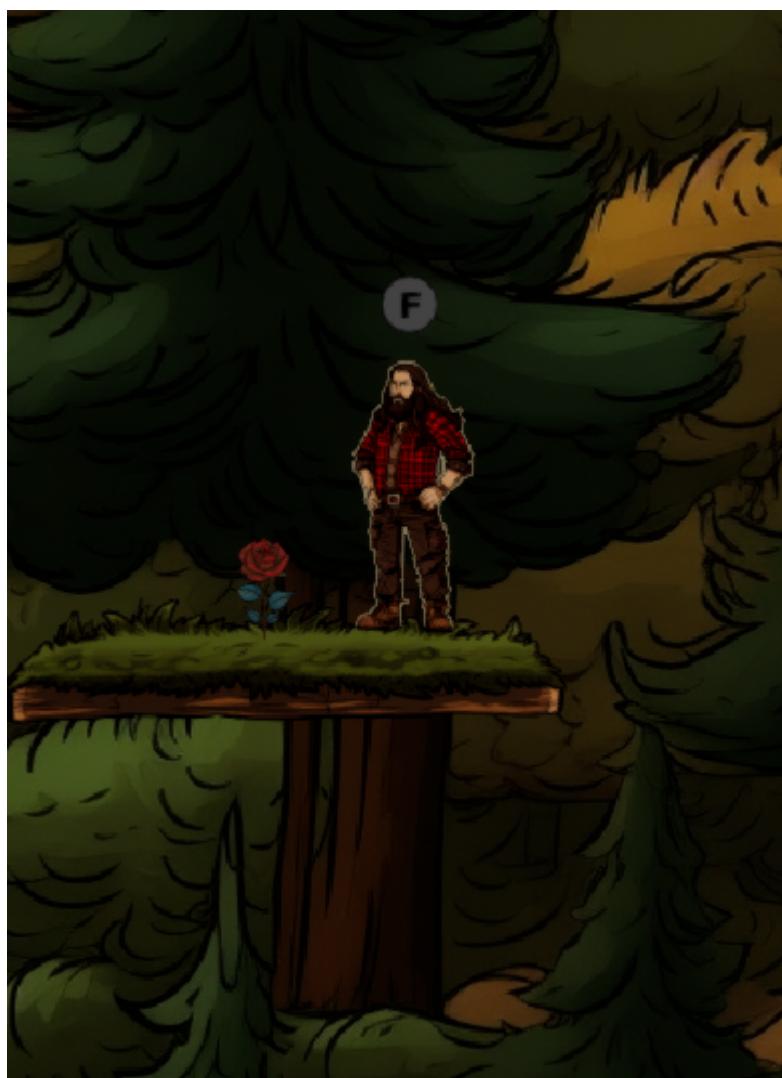
е нужно за катерени по новите летящи платформи, които среща. Героят може да минава през платформите стига да не е стъпил върху тях. Ако е на платформа, то тя става солидна и вече не може да се минава през нея. За да слезе от платформа и да я направи отново проходлива героя трябва или да слезе от нея, чрез движение наляво или надясно, или да натисне клавиша “S” на клавиатурата, което веднага прави платформата пропусклива и той пропада през нея. След успешно преминаване през платформите и капаните на тях, се стига до края на нивото.



Фиг. 38: Героят стой на летяща платформа

Четвърто ниво

В четвърто ниво героят отново се среща с мистериозният човек, който му казва да се оглежда за червено цвете. След преминаване през още платформи и препятствия на тях се стига до разклонение на пътя. Надясно е знака за край на ниво, но пред него стой невидима стена. Наляво има платформа с прасе и още една платформа, с цветето на нея, която е твърде далеч, за да се стигне до нея с обикновен скок. За да се стигне до нея героят трябва да се качи на прасето и да използва неговата инерция за да направи по дълъг скок. След взимане на цветето, невидимата стена изчезва и нивото може да се завърши.



Фиг. 39: Героят е напът да вземе цветето

Пето ниво

Нивото започва с още един разговор с мистериозният човек, в който героят му дава цветето и в замяна той подобрява обувките. Сега вече чрез натискане на клавиша “E” може да се сменя режима на скачане. Новият режим заменя двойния скок с един силен скок. След това героят се сблъска с нов противник. Това са катерици, които хвърлят жъльди. Героят също получава умението да избягва тези жъльди. Това става чрез натискане на “Shift”, което за кратко кара всеки жъльд да пропусне героя. По време на избягване той не може да се движи. При докосване на катерица героят я гони от нивото. Друго ново нещо в това ниво са летящите тръни. Те са на различни места из нивото и пречат на високи скокове.



Фиг. 40: Главният герой избягва жъльд хвърлен от катерица

Шесто ниво

В шесто ниви играчът се изправя срещу много катерици. Те имат нова вариация, която пуска жълъди от високо вместо да ги хвърля настрани. Има и нова вариация на летящите платформи, която се движи. След преминаване на първата група препятствия героят вижда няколко плодови храста. Плодовете на тези храсти възстановяват живот при изяддане. Във втората част на нивото героят има избор. Той може или да продължи надясно към края на нивото, или да тръгне наляво, където има трудни за минаване препятствия, но накрая има още едно цвете.



Фиг. 41: Героят стой пред до плодни храсти

Седмо ниво

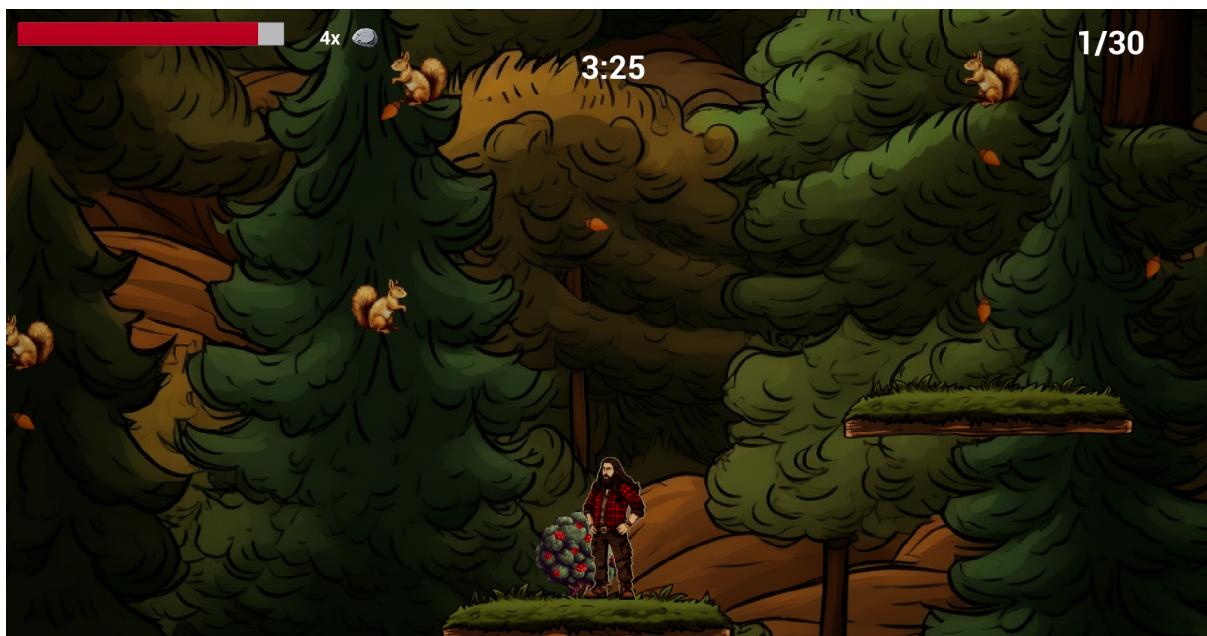
В началото на нивото мистериозният човек чака героят. Този разговор с него се е различен в зависимост от това дали героят е взел цветето от шесто ниво. Ако не е, мистериозният човек му казва да се оглежда по- внимателно и му пожелава късмет, след което си тръгва. Алтернативно героят му дава цветето и получава избор какво подобрение иска. Първият вариант е още едно подобрение на обувките, което позволява на героя да спринтира за кратко време. Това става чрез бързо натискане 2 пъти на бутона за движение на ляво или дясно. Другият вариант е отвара, която увеличава лечебните способности на плодовите храсти. След разговора героят среща нова опасност. Под земята има змии, който когато усетят, че героят е близо издават съскащ звук и малко след това изскочат. Остатька от нивото се състий от много препятствия.



Фиг. 42: Змия, която е изскочила от земята

Осмо ниво

Това ниво е последното и като такова е различно от другите нива. Вместо препятствия и знак на края на нивото, това ниво съдържа битка. В началото на нивото мистериозният човек обяснява на героя, че за да продължи трябва да победи "Съвета на катериците". Също така героят получава възможност да носи и хвърля камъни. Те се взимат от купчините с камъни и броят налични камъни може да се види до живота на героя. След продължаване надясно се стига до мястото, където ще се води битката. Когато героят стигне до средата на арената битката започва. Групи катерици започват да се появяват из арената и героят трябва да изгони всички 30 преди 4 минутния таймер да свърши. Играчът трябва умело да използва новото умение за хвърляне на камъни, за да успее. При победа над катериците на дъното на арената се появява цвете, което служи за край на нивото.



Фиг 43: Героят по време на битката

Заключение

Използването на дифузен модел за създаване на графиката на игра е възможно и крайният продукт изглежда добре, но все пак има и ограничения. Първото голямо ограничение е че играта трябва да е двуизмерна. При триизмерни игри настоящото поколение дифузни модели не могат да свършат работа. Друго голямо ограничение е липсата на анимации базирани на смяна на кадри. Това може да се заобиколи частично с анимации базирани на локално преместване и ротации на графичните елемент каквито има в играта, но те са с по-ниско качество от стандартните анимации. Също е важно да се отбележи, че ако играта имаше нужда от изображения на сложни машини, като например катапулт, дифузните модели не се справят добре с рисуването им. От страна на време за създаване на графиката, то варира значително в зависимост от сложността на модела и количеството приемливи дефекти. За нещо като скалата, използвана като разделител на частите в някой от нивата, тя отне само 5-10 минути за създаване обработка и интеграция, но по-важни изображения, като главния герой отне над 6 часа, поради нуждата от високо качество. Повечето обекти отнеха между 1-2 часа за генерация, избор и обработка. Голяма сила на Midjourney е създаването на фон. Почти едно на 5 изображения изглежда достатъчно добре, за да се използва. Допълнително на това има настройка, която прави изображението да може да се свързва незабележимо със себе си от всяка страна.

В края сметка дифузните модели изглеждат като приемлив начин за създаване на графика за игри, които се разработват от един човек или малък екип, когато няма наличен графичен художник. Най-практична за момента изглежда система, в която дифузен модел генерира базовото изображение и то се дава на графичен художник, за да се поправят дефектите. Така може да се намали значително времето нужно за създаване на изображение, тъй като, ако се разчита само на дифузния модел, то доста добре изглеждащи изображения с дефект трябва да се пропуснат.

Разбира се, това може да се промени в близкото бъдеще, като се има на предвид бързият темп на подобрене и иновации в сферата на изкуствения интелект.

Бъдеще на играта

За бъдещето на играта създадена за тази дипломна работа има създадени планове за нови нива, противници, продължаване на историята и добавяне на музика, за

създаването на която отново ще се използва изкуствен интелект. Също така ще се следят иновациите в дифузните модели, за потенциално разработване на модел, който може да анимира. Графиката ще си остане напълно създадена от Midjourney. Също така след завършване на пълната игра тя може да се дистрибутира на всяка платформа в благодарение на факта, че тъй като не използва никакъв външен код, Unreal Engine може да я компилира за всяка от тях.

Библиография

Сайт на Unreal Engine - <https://www.unrealengine.com>

Сайт на MidJourney - <https://www.midjourney.com>

Сайт на GIMP - <https://www.gimp.org>

Сайт на Sony Vegas - <https://www.vegascreativesoftware.com>

Форум за Unreal Engine въпроси - <https://forums.unrealengine.com>

Youtube канал “Mathew Wadstein” с много полезни видеа за Unreal Engine -
<https://www.youtube.com/@MathewWadsteinTutorials>