МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук,
управління та адміністрування
Кафедра інформаційних технологій

Бакалаврська кваліфікаційна робота

на тему: Розробка ігрового додатку на платформі
Game Maker Studio

Виконав студент 2 курсу групи К–25
Спеціальність 122 комп'ютерні науки
Гурін Юрій Владиславович

Керівник асистент
Бучинська Ірина Вікторівна

Консультант к.геогр.н., доцент
Коваленко Людмила Борисівна

Рецензент к.т.н., доцент
Гнатовська Ганна Арнольдівна

Одеса 2020

ЗМІСТ

Список скорочень, умовних позначень і термінів 5
Вступ 6
1 Аналіз предметної області гри 7
1.1 Визначення ігор та їх види 7
1.1.1 Види ігор 7
1.1.2 Визначення відеогр за місцем ієрархії ігор 12
1.1.3 Розробка відеоігор 15
1.2 Вимоги до програмного виробу 19
1.3 Вимоги до функціональних характеристик 19
1.4 Дослідження про користь ігор 21
2 Розробка технічного проекту 23
2.1 Обґрунтування вибору мови програмування 23
2.1.1 Види мов програмування які використовуються для програмування ігор . 23
2.1.2 Чому для написання програми був обраний саме С++ 25
2.2 Обґрунтування вибору середовища розробки 29
2.2.1 Опис ігрового движка GameMaker: Studio 29
2.2.2 Особливості ігрового движка GameMaker: Studio 30
2.2.3 Переваги й недоліки ігрового движка 32
2.2.4 Аналіз Game Maker Language 34
2.2.5 Аналоги Game maker studio 37
3 Реалізація технічного проекту 40
3.1 Розробка програмної системи 40
3.2 Опис геймплея 45
3.3 Опис сюжету та персонажів 48
Висновки 51
Перелік посилань 52

СПИСОК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

IDE – Integrated Development Environment − Інтегроване середовище розробки − система програмних засобів, яка використовується програмістами для розробки програмного забезпечення.

LC – Downloadable content форма поширення офіційного додаткового цифрового медіа контенту (мультимедійних файлів) через Інтернет.

GMS2 – GameMaker Studio 2 − двигун на якому робляться відеоігри різних жанрів у 2D розмірі.

GML – мова програмування яка інтегрована до двигуна (GameMaker Studio 2) для роботи безпосередньо у двигуні.

Двигун – базове програмне забезпечення комп’ютерної гри.

Ігровий процес, або геймплей (англ. Gameplay), – компонент гри, який відповідає за інтерактивне взаємодія гри і гравця. Геймплей описує, як гравець взаємодіє з ігровим світом, як ігровий світ реагує на дії гравця і як визначається набір дій, який пропонує гравцеві гра. Термін частіше вживається в контексті комп'ютерних ігор.

Штучний інтелект – розділ комп’ютерної лінгвістики та інформатики, що опікується формалізацією проблем та завдань, які подібні до дій, які виконує людина.

Game Maker: Studio – двигун для розробки відеоігор. (попередня версія двигуна GameMaker Studio 2.

ВСТУП

На сьогоднішній день ігрова індустрія дуже швидко розвивається та надає все більший стандарт якості. В умовах використання сучасних інформаційних технологій є все більше змог, щоб зробити дійсно гідний продукт, який сподобається аудиторії.

Game Maker − це неймовірно простий ігровий движок, що дозволяє створити гри для великого числа платформ − Windows, Mac OS X, Ubuntu, Android, iOS, Tizen, Windows Phone, Windows 8, PlayStation 3, PS 4, PS Vita, Xbox One і HTML 5. Є підтримка SteamWorks.

Швидкість розробки навіть при скромних знаннях і мінімальної мотивації суб'єктивно швидше, ніж на інших двигунах. Установка і налаштування для початківців максимально проста і не вимагає особливих знань. Компіляція під інші платформи не вимагає зміни коду гри і здійснюється одним кліком.

Мета кваліфікаційної роботи − розробити ігровий додаток за допомогою середовища Game Maker: Studio.

Об'єктом дослідження є розробка комп'ютерної ігор.

Предмет дослідження: технології розробки комп'ютерної.

Для досягнення поставленої мети необхідно виконати наступні завдання:

1) вибрати жанр, вид і платформу для комп'ютерної гри;

2) розробити сценарій, концепцію основних елементів;

3) вибрати і вивчити засіб реалізації;

4) підготувати необхідні для гри анімації;

5) реалізація додатку гри.

Кваліфікаційна робота містить вступ, 3 глави, висновки, 11 рисунків, 7 посилань.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ГРИ

1.1 Визначення ігор та їх види

Гра – діяльність людини з моделювання іншого виду діяльності з розважальною чи навчальною метою. Гра відрізняється від роботи тим, що не ставить перед собою безпосередньо корисної мети (хоча сама гра може мати свою власну корисність), а також тісно межує з мистецтвом, хоч зазвичай не створює художніх цінностей.[1][1)]

Перші ігри з'явилися у тварин задовго до виникнення людей. У мавп є не тільки соціальні ритуали, наприклад, шлюбні, а й ігри, схожі на людські. Людство грається і грає в ігри з доісторичних часів, починаючи з ритуальних ігор (наприклад, обряду ініціації), а з розвитком цивілізації ігри робилися дедалі складнішими й відобразили практично всі сфери життя суспільства: війну, кохання, історію тощо. Ймовірно, найскладніші нині ігри – MMORPG з режимом мультиплеєра на кшталт World of Warcraft, до якого щохвилини підключені тисячі користувачів у всьому світі, й у якому щомиті відбуваються мільйони різних дій.

1.1.1 Види ігор

Дитячі ігри – Людина грається змалечку. Важливим елементом дитячих ігор є іграшка. Дитячі ігри сприяють вивченню світу, розвитку особистості та уяви, усвідомленню дорослих ролей, виробленню навичок і встановленню спілкування між однолітками. Граючись, діти готуються до дорослого життя, вдаючи, що вони вчителі, лікарі, вихователі, влаштовують уявні вечірки своїм

[1)] [1] Йоган Гейзинга. Homo Ludens. Досвід визначення ігрового елемента культури.: Издательство Ивана Лимбаха.: 2016. 416 с.

лялькам тощо. Сучасна нейронаука вважає, що гра розвиває гнучкість розуму, виробляє навички й здатність досягати бажаної мети різними способами.

Часто дитячі ігри структуровані – мають встановлені правила, відомі всім учасникам. Такі ігри різні для різних народів, тобто мають національні й культурні особливості. Серед прикладів ігор, в які граються діти в Україні: хованки, штандер, цюці-бабки, класи, скакалка, гумки, ниточка, бачу тощо.

Для створення умов для дитячих ігор у міських житлових районах влаштовуються дитячі ігрові майданчики. Зазвичай на ігрових майданчиках облаштовуються пісочниці, гойдалки, драбинки й подібні структури, що допомагають дітям розвивати координацію й спритність, водночас просто отримуючи задоволення. Розважальні центри, аквапарки й інші комерційні заклади пропонують дітям складніші типи ігор, що потребують додаткового устаткування.

Салонні ігри – У салонні ігри грають здебільшого дорослі на вечоринках з метою весело провести час. Прикладами салонних ігор можуть бути вікторини або шаради, буріме тощо. На відміну від дитячих ігор, салонні ігри зазвичай не вимагають спритності й рухливості. Такі ігри сприяють виникненню невимушеного спілкування між гостями, подоланню скованості тощо. Іноді, за домовленістю, гравець при програші, або при порушенні правил, зобов'язаний виконати якийсь жартівливий фант.

Симуляції – До цього класу належать ігри, в яких створюється певна умовна ситуація з метою тренування, наприклад військові маневри. До цього класу ігор належать також різноманітні комп'ютерні стимулятори, що намагаються якомога точніше відтворити умови реальних ситуацій і навчити користувача приймати рішення в екстремальних умовах.

Одним із видів симуляції є бізнес-симуляція. Цей різновид симуляції характеризується імітацією бізнес-процесів в окремо взятому підприємстві, галузі, господарстві.

Спортивні ігри – Спортивні ігри мають добре встановлені правила, зафіксовані в документах відповідних міжнародних або національних федерацій, що ними опікуються. Аматори зазвичай притримуються цих правил. Спортивні ігри діляться на індивідуальні й командні. У командних іграх важливе не тільки вміння окремих гравців, а й вміння успішно взаємодіяти між собою.

Характерною особливістю спортивних ігор є те, що вони переросли рамки простої розваги, й стали не тільки частиною культури, а й індустрією. У спорті є професіональні гравці, для яких гра – це робота, а не розвага, й глядачі, вболівальники, які отримують задоволення, спостерігаючи за грою, і спортивний бізнес, що організовує ігри. Серед найпопулярніших спортивних ігор: футбол, теніс, гольф, баскетбол, волейбол, хокей, крикет, бейсбол.

Інші ігри, такі як алтимат, хоча не мають великої популярності у глядачів, приваблюють багатьох любителів. Багато ігор популярні тільки в межах окремих країн, як наприклад, скраклі.

Логічні ігри – До логічних ігор належать: головоломки, кросворди, судоку, а також багато з антагоністичних настільних ігор. Цікавість логічних ігор для людини в тому, що гра перед нею ставить задачу, розв'язок якої можна знайти за допомогою логічного аналізу, продумування на кілька кроків уперед, і успішне розв'язання задачі приносить людині задоволення.

До цієї категорії також можна віднести картярські ігри різного роду, оскільки вони вимагають логічного мислення, й у них часто грають для задоволення. Однак, картярські ігри також відносяться до азартних, бо у них часто грають на гроші, а іноді й до спортивних – такі ігри як покер, у деяких варіантах, і бридж мають правила, за якими можна проводити змагання.

Хоча за своєю суттю гра є людською діяльністю, що присить задоволення, логічні ігри часто розглядаються з математичної точки зору. Математичний аналіз багатьох логічних ігор допомагає знайти виграшну стратегію.

Комп'ютерні алгоритми на початок XX століття, здебільшого переважають людину в більшості ігор. Тільки для бриджу та покеру, наприклад техаського безліміту, алгоритми поки що поступаються людському мисленню. З іншого боку, розробка алгоритмів гри сама є цікавою грою.

Настільні ігри – проводяться з використанням певних ігрових об'єктів на площині, часто – на спеціальній дошці. Серед популярних настільних ігор: шахи, шашки, нарди, го тощо. Зазвичай це антагоністичні ігри зі складною стратегією.

Найчастіше настільні ігри розраховані на двох гравців, але бувають ігри з кількома гравцями, такі як монополія, маджонг, доміно, лото чи бінго. У деяких іграх, наприклад бриджі, гравці утворюють команди. До настільних ігор належать також рольові ігри, які здебільшого не мають антагоністичного характеру, і в них грають для задоволення та спілкування між гравцями.

Азартні ігри – Відмінністю азартних ігор є те, що виграш чи програш у них має не тільки символічне, а й реальне життєве значення – вони граються на гроші. Відповідно виграш і програш у них здебільшого залежить не тільки від майстерності гравця, а й від випадку. В сучасному світі в одних країнах існує індустрія азартних ігор, в інших країнах азартні ігри знаходяться під жорстким контролем або заборонені.

Найпростіші азартні ігри – ігри випадку – не потребують вміння гравців. У них виграш чи програш залежить від результату випадкової події, наприклад орел чи решка при підкиданні монети або випадання певного сектора в грі в рулетку. Однак, складніші ігри, навіть з такими простими правилами як блекджек вимагають від гравця розумової дисципліни й знання ймовірностей отримання тієї чи іншої комбінації карт. У ще складніших іграх у карти, покері, преферансі, вміння гравця набагато важливіше ніж випадковість роздачі карт – вмілий гравець отримає максимальний виграш на хорошій карті й мінімізує програш на поганій.

У країнах, де гральний бізнес дозволений, існують спеціальні заклади для гри – казино. Для простіших ігор існують гральні автомати. Привабливість азартних ігор не тільки задоволення від процесу гри, а й прагнення гравців швидко розбагатіти. Гральний бізнес використовується також для відмивання грошей. З азартними іграми пов'язана також така проблема як лудоманія – психічний розлад, при якому гра витісняє в житті людини інші цінності.

Відеогра – Ігри із застосуванням електронних пристроїв, як ігрові автомати, гральні приставки чи персональні комп'ютери. Окремий різновид електронних ігор – відеоігри, в яких гравець керує візуальними образами на екрані. Гравців у відеоігри зазвичай називають геймерами. Те, що геймер робить, граючи у відеогру, і те, як комп'ютерна система відповідає на дії гравця, часто називають геймплеєм.

У відеоіграх правила закладені в комп'ютерній програмі, а супротивником виступає або сама програма або інші люди. Швидкий розвиток відеоігор призвів також до появи кіберспорту. Популярність відеоігор породила також таке негативне явище, як залежність від них. За своєю природою гра повинна приносити задоволення, але надмірна погоня за рекордними показниками стає для залежних гравців штучною метою, що витісняє з життя інші цінності.

Із появою та розповсюдженням такого явища, як Інтернет, з'явилося й поняття онлайн-ігор. Такі ігри є основою комунікації між людьми, об'єднаними спільними інтересами, незалежно від їхньої національності та місця проживання. Найбільш розповсюдженими і успішними вважаються браузерні ігри, та настільні онлайн ігри, зокрема: нарди, шахи, покер тощо. До онлайн-ігор належать також багатокористувацькі онлайнові рольові ігри, що приваблюють тисячі й мільйони гравців зі всього світу.

Рольові ігри – У рольових іграх гравці діють у певному, заданому правилами, уявному світі, беручи на себе певні вигадані ролі. Рольові ігри часто не носять змагального характеру, а призначені для того, щоб отримати

задоволення від самого процесу гри. Тому в них часто за правилами переможець і переможений не визначається.

Рольові ігри поділяються на ігри живої дії, в яких гравці безпосередньо створюють ігрову ситуацію, настільні рольові ігри, в яких група гравців використовує або спеціальні ігрові набори, або ручку та папір, та комп'ютерні рольові ігри, в яких дія відбувається на екрані комп'ютера.

1.1.2 Визначення відеогр за місцем ієрархії ігор

Відеогра – це електронна гра, в ігровому процесі якої гравець використовує інтерфейс користувача, щоб отримати зворотну інформацію з відеопристрою. Електронні пристрої, які використовуються для того щоб грати, називаються ігровими платформами. Наприклад, до таких платформ належать персональний комп'ютер та гральна консоль. Пристрій введення, який використовується для керування грою, називається ігровим контролером. Це може бути, наприклад, джойстик, клавіатура та мишка, геймпад або сенсорний екран. [2][1)]

У 2011 році відеоігри були офіційно визнані видом мистецтва урядом США та Національним фондом мистецтв США. Однак, загальноосвітове визнання їх мистецтвом лишається дискутивним питанням.

Місце в ієрархії ігор:

Нерідко відеоігри в повсякденному житті та пресі називають комп'ютерними іграми. Проте розробники та дослідники таких ігор послуговуються терміном «відеогра», виходячи з їх специфіки та місця в ієрархії ігор загалом:

[1)] [2] Wolf M. J. P. Encyclopedia of Video Games: The Culture, Technology, and Art of Gaming Santa Barbara CA: Greenwood.: 2012. 740 p.

- гра – діяльність із розважальною і/або навчальною метою за встановленими правилами. Включає в себе такі різновиди як дитячі, спортивні, настільні, азартні, рольові та, окрім інших, електронні ігр;
- електронна гра – гра, що відбувається за допомогою електронних пристроїв, програмованих чи непрограмованих, які утворюють інтерактивну систему (такими можуть бути пінбольні автомати, слотові машини, електромеханічні ігри, лотереї, аудіоігри і відеоігри; оскільки відеоігри на програмованих пристроях складають основну їх частину, нерідко терміни «електронна гра», «комп'ютерна гра» і «відеогра» вживаються як тотожні);
- комп'ютерна гра – гра, яка забезпечується програмно керованим електронним пристроєм – комп’ютером, це також може бути слотовий автомат, пінбольна машина, аудіо чи відеогра;
- відеогра – гра, яка відбувається через керування візуальними образами на моніторі чи іншому дисплеї (може забезпечуватися як програмованим, так і не програмованим електронним пристроєм: аркадним ігровим автоматом, ігровою приставкою, персональним комп'ютером; зрідка під відеогрою розуміється гра тільки для ігрової приставки);
- гра для персонального комп'ютера – відеогра, яка запускається на персональному комп'ютері, часто називається просто комп'ютерною грою, але не тотожна комп'ютерній грі, описаній вище.

Історія появи відеоігор передували розвиток програмованих комп'ютерів та технологій формування зображення на екранах електронних пристроїв. Різноманітні електронні та механічні ігрові пристрої існували ще в першій половині XX століття, але не мали досить значного поширення. Попередниками відеоігор є пристрій «Cathode ray tube Amusement Device» (укр. розважальний пристрій з електронно-променевою трубкою), патент на яку Томас Ґолдсміт

Молодший (англ. Thomas T. Goldsmith Jr) та Істл рей Менн (англ. Estle Ray Mann) отримали 14 грудня 1948 року, і шахова комп'ютерна програма, розроблена у 1947 Аланом Тьюрінгом. Початково ігрові програми, як шахи чи хрестики-нулики, розроблялися в рамках військової програми США у прагненні створити комп'ютер, здатний передбачати дії противника.

Перша успішна спроба створити розважальний пристрій, який використовує для зворотнього зв'язку із гравцем відео, належить Вільяму Гіґінботаму. У 1958 він розробив Tennis For Two, однак не розглядав гру як щось важливе і зрештою розібрав обладнання для інших, наукових, проектів.

В 1960-ті студентами Массачусетському технологічному інституту було написано гру Spacewar!, яка у 1966 підштовхнула Sanders Associates на думку про можливість створення грального пристрою, що під'єднувався б до домашніх телевізорів. Коли комп'ютери стали порівняно дешевими і могли використовуватися не лише науковими установами, студент Стенфорду Білл Піттс, вражений Spacewar, вирішив, що на основі комп'ютера PDP–11 реально створити пристрій спеціально для ігор – Galaxy Game. Разом з другом Х'ю Таком він розробив ігровий автомат, що за монети надавав можливість пограти і таким чином окупив би себе. За підтримки Нолана Бушнелла з компанії Nutting Associates цей автомат, Computer Space і однойменна гра на ньому 1971 року, став першим комерційним пристроєм з відеогрою. Як екран був використаний чорно-білий телевізор, комп'ютерною основою виступили мікросхеми TTL серії 7400. Вже у 1972 році з'явилась перша домашня гральна консоль, Magnavox Odyssey, яка підключалася до звичайного телевізора. Поштовхом до виникнення індустрії відеоігор стала висока популярність спочатку аркадної (1972), а потім і домашньої відеогри Pong (1975), оскільки її комерційний успіх призвів до появи великої кількості клонів від інших компаній.

1.1.3 Розробка відеоігор

Розробка відеогри має низку послідовних етапів, загалом їх є три: розробка програмного (джерельного) коду, розробка контенту (малюнки, моделі, музика) та розробка ігрових механік. Їм передує проектування (пре-продакшну) – генерування геймдизайнером ідей щодо майбутньої гри, вибір жанру, тематики, особливостей ігрового процесу, розробка сценарію та образів персонажів з оточенням. Менеджер координує дії різних людей, залучених до розробки, складає план їхньої роботи, встановлює терміни її виконання, планує витрати. Готова гра в свою чергу має пройти низку етапів, в ході яких потрапляє до гравців і підтримує інтерес до себе. Індустрія відеоігор включає у себе багато людей з різними професіями та ролями: програмістів, які відповідають за технічні можливості гри, художників, моделювальників та аніматорів, які створюють графічний контент, композиторів та звукорежисерів, які створюють звукове оформлення та музичний супровід, який нерідко видається окремим накладом. За успішне завершення роботи над проектом відповідають продюсери. Відеоігри, які розробляються незалежними розробниками чи аматорами називаються інді-іграми. Такі ігри нерідко створюються за допомогою спеціальних програм, які можуть не вимагати окремо розробки коду або графіки, наприклад, як RPG Maker.

Написання коду є основою будь-якої відеогри і відповідає за її технічні можливості, від яких залежить контент та ігровий процес. Сучасні ігри здебільшого засновані на готових програмних модулях – ігрових рушіях, де вже реалізовані базові функції, здатні зв'язувати воєдино графіку, звук, об'єкти і їх рухи. Однак програмісти все-одно мусять писати код, щоб налаштувати рушій і сповна реалізувати задуману гру. Деякі розробники створюють власні рушії для конкретної гри. Існують як вільні ігрові рушії, доступні будь-кому, так і ті, що

вимагають отримання ліцензії на їх використання. Останні як правило володіють ширшим функціоналом і використовують передові технології.

Розробка контенту: вигляд персонажів та навколишнього віртуального світу початково розробляється через вибір з-поміж кількох варіантів. Ці початкові малюнки або тривимірні моделі називаються концепт-артами. На їх основі створюються двовимірні або тривимірні моделі, які вже будуть використовуватися в грі. В програмах анімації задаються їх рухи, пози. Для реалізації реалістичних рухів чи емоцій може застосовуватися захоплення руху живих акторів. Візуальні ефекти роблять гру видовищнішою і задають стиль: відкидання тіней, заломлення світла, постріли і вибухи, стилізація під малюнок або кіноплівку. Дизайнер рівнів розміщує готові об'єкти в ігровому світі та продумує їх рухи. За реалізацію картинки і звуку відповідають графічний і звуковий рушії.

Для звукового оформлення пишеться музика і відбувається озвучування персонажів. Крім того для повноцінного звукового оформлення потрібні ефекти, як кроки, звуки пострілів. Вони можуть обиратися з вільних бібліотек чи записуватися окремо. Часом ігри містять відеовставки, створені в програмах двовимірної чи тривимірної анімації. Інколи для відеовставок знімаються живі актори і будуються декорації. Є актори, які спеціалізуються саме на зйомках в таких відеовставках або озвучуванні персонажів. Сюжет, діалоги, додаткові тексти пишуться сценаристами і відповідальними за це письменниками.

Розробка ігрової механіки визначає насиченість ігрового процесу, правила, за якими грається відеогра. Основою механіки є ігрові об'єкти, такі як персонажі, об'єкти, з якими вони можуть маніпулювати, декорації. Частиною ігрової механіки є управління, якими чином гравець керує персонажем та ігровим світом. Наприклад, як задається напрям руху, як активізується взаємодія з віртуальними предметами. Крім того на етапі розробки механіки створюється користувацький інтерфейс, який інформує гравця і дозволяє

взаємодіяти зі світом гри. За реалістичну взаємодію об'єктів відповідає фізичний рушій. До прикладу, він реалізує закони інерції, гравітацію, поведінку рідин, властивості предметів. Штучний інтелект (ШІ) відповідає за поведінку персонажів, як вони реагуватимуть на дії гравця.

Тестування після завершення праці над кодом, контентом і механікою, за яких гра може функціонувати, відбувається її доопрацювання. Гра, не зібрана до кінця, але в яку можливо грати, називається альфа-версією. Вона може містити значні помилки і не доопрацювання, як відсутність певних можливостей, музики або об'єктів. Виявленням проблем займаються тестери, котрі грають в цю гру, намагаючись сповна скористатися всіма доступними можливостями в ній. Зазвичай на цьому етапі розробники записують рекламний трейлер, показуючи ігровий процес на відео, даючи потенційним гравцям уявлення про свою роботу. На пізнішому етапі виходить бета-версія, до тестування якої можуть залучатися і потенційні покупці гри. В бета-версії відбувається подальший пошук помилок, перевірка коректності взаємодії об'єктів ігрового світу, управління. Можливі внесення змін в оформлення, зміна ігрового балансу, можливостей персонажів.

Випуск і продажі як правило відповідає видавець гри, від якого розробник отримує частку отриманих коштів. В разі якщо розробник і видавець є частинами однієї компанії, розподіл прибутків відбувається за встановленими там правилами. Видавець має піар-менеджерів та ігрових журналістів, котрі певним чином заявляють про існування гри, проводять рекламну кампанію, повідомляють про неї, викладаючи відео, публікуючи новини в спеціалізованих виданнях, демонструючи гру на виставках відеоігор. Видавець займається локалізацією гри, тобто перекладом її тексту і озвучування, адаптацією до законів і культури тої країни, де гра видаватиметься. Наприклад, в деяких країнах заборонено зображення крові або оголеного тіла, відповідно такі зображення доводиться видаляти з гри. Для локалізації існують спеціальні

команди фахівців, які крім власне перекладу роблять гру зрозумілою для гравця конкретної країни. В сучасних іграх текст і звуки як правило містяться в окремих файлах, що робить локалізацію простішою. Існують офіційні і неофіційні локалізатори. Часто гра від початку має локалізації на найбільш поширені мови, як англійську, іспанську і китайську. Іншими мовами гра офіційно виходить згодом чи цим неофіційно займаються ентузіасти.

Завершена і локалізована гра записується на певні носії, як DVD-диски, або надається для завантаження з Інтернету. Диски поставляються в магазини, а версії для завантаження поширюються через сервіси цифрової дистрибуції, як Steam. Цифрова дистрибуція, на відміну від дисків, іноді передбачає і безкоштовне завантаження гри. Так декотрі ігри, які представляють вже тільки історичний інтерес, надаються безкоштовно. В сервісах електронної дистрибуції часто відбуваються акції, які надають знижки аж до 100 %.

Підтримка після виходу гри в ній можуть виявитися досі невідомі помилки програмного забезпечення, не виявлені тестувальниками під час розробки. Це може статися тому, що збій виникає лише за незвичних обставин у грі, вважається незначним, або тому, що розробка гри була занадто пришвидшеною для того, щоб задовольнити термін публікації. Такі помилки можуть варіюватися від незначних графічних хиб до серйозних, які можуть ускладнювати чи взагалі робити неможливим грання. Переважно видавці випускають оновлення (патчі) для виправлення помилок. Іноді помилки можуть виявлятися корисним для гравців, дозволяючи обходити встановлені правила.

Гравців може не влаштувати певний аспект гри, до прикладу, складність. Розробники в такому разі також випускають патчі, які додають певні вдосконалення. Розробка і випуск патчів не приносять прибутків самі по собі, але можуть збільшити продажі гри, оскільки вона позбавляється недоліків.

Для підтримання інтересу до гри до неї випускаються доповнення, що крім виправлення помилок додають нові можливості чи доповнюють і

розвивають сюжет. Доповнення бувають як платні, так і безкоштовні. Ті доповнення, що розповсюджуються через Інтернет, називаються DLC (Downloadable content). Переважно DLC є платними і не змінюють гру суттєво, але надають нові можливості, оформлення персонажа.

1.2 Вимоги до програмного виробу

Вимоги до програмного забезпечення – набір вимог щодо властивостей, якості та функцій програмного забезпечення, що буде розроблено, або знаходиться у розробці. Вимоги визначаються в процесі аналізу вимог та фіксуються в специфікації вимог, діаграма прецедентів та інших артефактах процесу аналізу та розробки вимог.

Розробка вимог до програмної системи може бути розділена на декілька етапів:

- знаходження вимог (збір, визначення потреб зацікавлених осіб та систем);
- аналіз вимог (перевірка цілісності та закінченості);
- специфікація (документування вимог);
- тестування вимог.

1.3 Вимоги до функціональних характеристик

Гра повинна розважати гравця, та надавати йому позитивних емоції. Також вона повинна мати функцію реіграбельность. Реіграбельность означає різні речі для різних людей, як я вже говорив. Деяким популярним авторам з розробки ігор не подобається термін, тому що він настільки розпливчасті, що може заплутати. Індустрія розробки ігор досить молода і її жаргон як і раніше розширюється. Наприклад, мореплавство досить стара річ і його жаргон визрівав кілька століть. Є багато-багато способів висловити принципи мореплавства. Розробці ігор потрібно ще кілька століть, щоб жаргон розвинувся

і зміцнів. Проте, у нас немає терпіння старих моряків, ми, розробники ігор – група швидких людей, які намагаються висловити поняття в цьому виді мистецтва, поки наука і теорія досі грають в догонялки.[3][1)]

Поговоримо про двох можливих підходах до реіграбельності. Перший – дії по поверненню в гру, поки подорож не буде завершено. Це про притягнення гравців, щоб вони не забували гру. Наша мета – викликати почуття, щоб змусити їх думати «ще один рівень і все». Є багато способів, як викликати таку поведінку і деякі найбільш поширені методи повинні бути відомі всім:

- кращі результати. Ігри з таблицею лідерів, куди гравець буде постійно повертатися, щоб переконатися, що його власний рекорд не побитий або подивитися топ інших гравців. Це стало прямо змаганням з появою всюдисущої мережі.
- кілька рівнів − прогрес в грі через різні рівні; щоб перемогти, гравець повинен завершити кожен рівень, це поклик до мандрівникові всередині кожного з нас, гравці продовжують грати, тому що вони повинні закінчити подорож.
- колекціонування та досягнення − деякі ігри пропонують досягнення і колекціонування всередині гри; люди продовжують грати, щоб зібрати все; якщо пункт вище звертається до нашого внутрішнього мандрівникові, то цей звертається до нашого внутрішнього скнарі, досягнення в поєднанні з соціальними мережами є чудовою річчю для залучення гравців.
- Сюжет − у деяких іграх важливий сюжет і люди будуть продовжувати грати, тому що хочуть дізнатися, що відбувається; це, як правило, поєднується з рівнями; квести були вершиною ігор з сюжетом; більшість сучасних 3D-ігор на приставках мають сюжет; іноді сюжет

[1)] [3] Розробка гри на JavaScript. URL: https://webref.ru/mobile/firefoxosgame/levels (дата звернення 23.05.2020).

недостатньо хороший або не має сенсу, але все ж ми всі ненавидимо незакінчених історій, тому продовжуємо грати.

Ці підходи, як правило, комбінуються. Наприклад, часто можна побачити таблицю з результатами і списки досягнень. При проектуванні гри ви повинні подумати про те, як залучити гравців і додати грі цінність.

1.4 Дослідження про користь ігор

Комп'ютеризація нашого життя вже стала незаперечним фактом і принесла з собою поряд з користю чимало проблем. Стосується це і до комп'ютерних ігор – тим більше, що ними в основному захоплюються діти і підлітки. У суспільстві все більше розповсюджується думка про те, що вони небезпечні для психіки дитини: розвивають в ньому агресивність, звужують коло його інтересів, збіднюють емоційну сферу. Ґрунтуючись на результатах досліджень ряду вітчизняних і зарубіжних фахівців, автор статті стверджує, що велика частина подібних уявлень помилкова. [4][1)]

Введення комп'ютерних ігор в структуру захоплень підлітка лише на кілька тижнів відволікає його від інших видів дозвілля, а потім більшість дітей повертаються до своїх колишніх хобі. Абсолютно протилежні розхожим дані отримані і щодо соціальної поведінки підлітків: гравці в комп'ютерні ігри легше, ніж контрольна група, адаптуються в компанії однолітків. Крім того встановлено, що проводять за комп'ютером свій вільний час підлітки в цілому більш кмітливі і цілеспрямовані, ніж їх однолітки, і краще оцінюють свої здібності і можливості.

Не було виявлено серед "гравців" ні серйозних порушень психіки, ні симптомів "комп'ютерної" залежності, хоча і серед них, як і серед любителів інших видів дозвілля, є фанати, число яких, однак, не перевищує 10 – 12%. В

---

1) [4] С. Шапкин. Компьютерная игра: новая область психологических исследований.: "Психологический журнал": 1999. том 20. № 1. С. 86-102.

основному надуманою виявилася і проблема зростання агресивності. І тільки у дітей до 9 років її в якійсь мірі можуть стимулювати гри з агресивним змістом, складові, втім, відносно невеликий відсоток серед загальної їх кількості.

Сьогодні під комп'ютерними іграми на увазі досить широкий клас програм і технічних засобів з різними можливостями: ігри-"діагности" і ігри-"терапевти", ігри-тренажери і навчальні ігри, і для кожної дитини треба в кожному конкретному випадку підбирати ті з них, які можуть принести йому максимальну користь.

Що ж стосується шкоди від комп'ютерних ігор, то він, головним чином, пов'язаний з непомірним їх використанням. Для будь-якого віку існують, як відомо, певні часові норми, і дотримуватися їх треба неухильно.

2 РОЗРОБКА ТЕХНІЧНОГО ПРОЕКТУ

2.1 Обґрунтування вибору мови програмування

2.1.1 Види мов програмування які використовуються для програмування ігор

C++ – компільований, статично типізований мова програмування загального призначення.

Підтримує такі парадигми програмування як процедурне програмування, об'єктно-орієнтоване програмування, узагальнене програмування, забезпечує модульність, роздільну компіляцію, обробку винятків, абстракцію даних, оголошення типів (класів) об'єктів, віртуальні функції. Стандартна бібліотека включає, в тому числі, загальновживані контейнери і алгоритми. C++ поєднує властивості як високорівневих, так і низькорівневих мов. У порівнянні з його попередником – мовою C, – найбільшу увагу приділено підтримці об'єктно-орієнтованого і узагальненого програмування.

C++ широко використовується для розробки програмного забезпечення, будучи одним з найпопулярніших мов програмування. Область його застосування включає створення операційних систем, різноманітних прикладних програм, драйверів пристроїв, додатків для вбудованих систем, високопродуктивних серверів, а також розважальних програм. Існує безліч реалізацій мови C++, як безкоштовних, так і комерційних і для різних платформ.

C# – об'єктно-орієнтована мова програмування. Розроблено в 1998–2001 роках групою інженерів під керівництвом Андерса Хейлсберг в компанії Microsoft як мова розробки додатків для платформи Microsoft .NET Framework і згодом був стандартизований як ECMA–334 і ISO / IEC 23270.

C# відноситься до сім'ї мов з C-подібним синтаксисом, з них його синтаксис найбільш близький до C++ і Java. Мова має статичну типізацію, підтримує поліморфізм, перевантаження операторів (в тому числі операторів явного і неявного приведення типу), делегати, атрибути, події, властивості, узагальнені типи і методи, ітератори, анонімні функції з підтримкою замикань, LINQ, виключення, коментарі у форматі XML.

Перейнявши багато від своїх попередників – мов C++, Pascal, Модула, Smalltalk і Java – C# виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад, C# на відміну від C++ не підтримує множинне успадкування класів (між тим допускається множинне спадкування інтерфейсів).

Python – є широко використовуваним мовою програмування загального призначення, високого рівня. Його філософія дизайну підкреслює читаність коду, а його синтаксис дозволяє програмістам, висловити поняття в меншій кількості рядків коду, ніж було б можливо в таких мовах, як C++ або Java. Мова забезпечує конструкції, призначені для того, щоб програми були чіткі на обох малих і великих масштабах.

Python підтримує кілька парадигм програмування, в тому числі об'єктно-орієнтованого, імперативний і функціональному програмуванні або процедурних стилів. Він має динамічну систему типів і автоматичне керування пам'яттю і має велику і всеосяжну стандартну бібліотеку.

Компілятори Python є для установки на багатьох операційних системах, що дозволяє виконувати код Python на найрізноманітніших систем. Використання сторонніх інструментів, таких як py2exe або Pyinstaller. Python код може бути зібраний в автономний виконуваний файл для деяких з найбільш популярних операційних систем, що дозволяє поширення програмного забезпечення Python основі для використання на цих середовищах, не вимагаючи установка інтерпретатора Python.

CPython, посилання реалізація Python, є вільне і відкрите програмне забезпечення і має модель розвитку громад, як це роблять майже всі його альтернативні реалізації. CPython управляється некомерційною компанією Python Software Foundation.

Microsoft Visual Basic – це мова програмування і інтегроване середовище розробки програмного забезпечення. Розроблявся компанією Microsoft. Visual Basic успадкував стиль і частково синтаксис мови BASIC, що має чимало діалектів. Visual Basic поєднує в собі процедури та елементи об'єктно-орієнтованих і компонентно-орієнтованих мов програмування, а інтегроване середовище розробки VB містить в собі інструменти для візуального проектування призначеного для користувача інтерфейсу, редактор коду з можливістю IntelliSense і підсвічуванням синтаксису і інструменти для налагодження додатків.

Delphi – імперативний, структурований, об'єктно-орієнтована мова програмування з суворою статичної типізацією змінних. Основна область використання – написання прикладного програмного забезпечення. Спочатку носив назву Object Pascal і історично сходить до однойменного діалекту мови, розробленого в фірмі Apple в 1986 році групою Ларрі Теслера. Однак в даний час термін Object Pascal найчастіше вживається в значенні мови середовища програмування Delphi. Починаючи з Delphi 7, в офіційних документах Borland стала використовувати назву Delphi для позначення мови Object Pascal.

2.1.2 Чому для написання програми був обраний саме C++

Тому що – Мова програмування C ++ була створений на початку 1980–х років, його творець співробітник фірми Bell Laboratories – Бйорн Страуструп. Він придумав ряд удосконалень до мови програмування C, для власних потреб. Т. е. Спочатку не планувалося створення мови програмування C ++. Ранні версії мови C ++, відомі під ім'ям «Сі з класами», почали з'являтися з 1980 року. Мова

С, будучи базовою мовою системи UNIX, на якій працювали комп'ютери фірми Bell, є швидким, багатофункціональним і стерпним. Страуструп додав до нього можливість роботи з класами та об'єктами, тим самим зародив передумови нового, заснованого на синтаксисі С, мови програмування. Синтаксис С++ був заснований на синтаксисі С.

Розвиток. Перше видання «Мови програмування С++» вийшло в 1985 році. Забезпечує перший опис цієї мови, що було важливо через відсутність офіційного стандарту. У 1989 році вийшла версія 2.0 мови С++. Його нові можливості включали множинне спадкування, абстрактні класи, статичні функції-члени, функції-константи і захищені члени. У 1990 році вийшло «Кухня країн довідкове керівництво по С++», покладене згодом в основу стандарту. Останні оновлення включали шаблони, виключення, простору імен, нові способи приведення типів і логічний тип.

У 1998 році був опублікований стандарт мови ISO / IEC 14882: 1998 (відомий як С ++ 98), розроблений комітетом з стандартизації С ++ (ISO / IEC JTC1 / SC22 / WG21 working group). Стандарт складається з двох частин – основи мови (core language) і стандартної бібліотеки мови, яка включає Standard Template Library (STL) і модифікований варіант стандартної бібліотеки мови С.

У 2003 році був опублікований стандарт мови ISO / IEC 14882: 2003, де були виправлені виявлені помилки і недоліки попередньої версії стандарту.

У 2005 році був випущений звіт Library Technical Report 1. Звіт описує розширення стандартної бібліотеки, які, повинні бути включені в наступну версію мови С ++.

З 2009 року велася робота по оновленню попереднього стандарту, попередньої версією нового стандарту спершу був С++ 09, а через рік С++ 0x, сьогодні – С++ 11, куди були включені доповнення в ядро мови і розширення стандартної бібліотеки.

Об'єктно-орієнтовані особистості C++ додає до C об'єктно-орієнтовані можливості. Він вводить класи, які забезпечують три найважливіші властивості ООП: інкапсуляцію, успадкування і поліморфізм.

Методи класу – це функції, які зможуть застосовуватися до екземплярів класу. Грубо кажучи, метод – це функція оголошена всередині класу і призначена для роботи з його об'єктами. Методи оголошуються в тілі класу. Описуватися можуть там же, але можуть і за межами класу (всередині класу в такому випадку досить представити прототип методу, а за межами класу визначати метод поставивши перед його ім'ям – ім'я класу і оператор). Методи і поля входять до складу класу називаються членами класу. При цьому методи часто називають функціями-членами класу.

Спадкування. У C ++ при спадкуванні одного класу від іншого успадковується реалізація класу, плюс клас-спадкоємець може додавати свої поля і функції або перевизначати функції базового класу. Множинне спадкування дозволено. Конструктор спадкоємця викликає конструктори базових класів, а потім конструктори нестатичних членів-даних, які є екземплярами класів. Деструкція працює в зворотному порядку. Спадкування буває публічним, захищеним і закритим.

Поліморфізм. Метою поліморфізму, стосовно об'єктно-орієнтованого програмування, є використання одного імені для завдання загальних для класу дій. Виконання кожного конкретного дії буде визначатися типом даних. Перевагою поліморфізму є те, що він допомагає знижувати складність програм, дозволяючи використання того ж інтерфейсу для завдання єдиного класу дій. Вибір же конкретного дії, в залежності від ситуації, покладається на компілятор. Поліморфізм може застосовуватися також і до операторів.

Інкапсуляція. Основним способом організації інформації в C++ є класи. На відміну від структури (struct) мови C, яка може складатися тільки з полів і вкладених типів, клас (class) C++ може складатися з полів, вкладених типів і

функцій-членів. Інкапсуляція в C++ реалізується через вказівку рівня доступу до членів класу: вони бувають публічними (public), захищеними (protected) і закритими (private). У C++ структури відрізняються від класів тим, що за замовчуванням члени і базові класи у структури публічні, а у класу – власні.

Гідності й недоліки. C++ – мову, що складається еволюційно. Кожен елемент C++ запозичувався з інших мов окремо і незалежно від інших елементів (ніщо з запропонованого C++ за всю історію його розвитку не було нововведенням в Computer Science), що зробило мову надзвичайно складним, з безліччю дублюються і взаємно суперечливих елементів, блоки яких засновані на різних формальних базах.

Критики C++ протиставляють йому який-небудь конкретний мову, а навпаки, стверджують, що для будь-якого випадку застосування C++ завжди існує альтернативний інструментарій, що дозволяє вирішити ту ж задачу більш ефективно і якісно. У свою чергу, прихильники C++ вважають некоректним порівнювати різні аспекти C++ з абсолютно різними мовами, так як загальний набір засобів і можливостей C++ істотно ширше, ніж в більшості мов, з якими проводиться порівняння, і сама по собі широта можливостей , на їх погляд, є вагомим виправданням недосконалості кожної окремо взятої можливості. Більш того, на їхню думку, висока сумісність з Сі є однією з важливих характеристик мови, і тому всі недоліки C++ виправдані перевагами, наданими цій сумісністю.

Гідності:

– висока сумісність з мовою Сі;
– обчислювальна продуктивність;
– підтримка різних стилів програмування: структурний, об'єктно-орієнтоване, узагальнене програмування, функціональне програмування, що породжує мета програмування;
– автоматичний виклик деструкторів об'єктів (в порядку зворотному виклику конструкторів) спрощує і підвищує надійність управління

пам'яттю і іншими ресурсами (відкритими файлами, мережевими з'єднаннями, т.п.);

– перевантаження операторів;

– шаблони (дають можливість побудови узагальнених контейнерів і алгоритмів для різних типів даних);

– можливість розширення мови для підтримки парадигм, які не підтримуються компіляторами безпосередньо;

– доступність. Для C++ існує величезна кількість навчальної літератури, перекладеної на всілякі мови.

Недоліки:

– погано продуманий синтаксис звужує спектр застосування мови;

– мова не містить багатьох важливих можливостей;

– мова містить небезпечні можливості;

– продуктивність праці програмістів на мові виявляється невиправдано низька;

– громіздкість синтаксису;

– важка спадщина;

– необхідність стежити за пам'яттю.

2.2 Обґрунтування вибору середовища розробки

2.2.1 Опис ігрового движка GameMaker: Studio

GameMaker: Studio – один з найпопулярніших ігрових движків, що дозволяє розробляти додатки під безліч платформ. GameMaker: Studio є серйозним розвитком його попередника – Game Maker і головною відмінністю є додавання платформ, завдяки якій, а також іншим істотним доопрацюванням, GameMaker: Studio став потужним інструментом для професійної розробки. Творець і головний розробник перших шести версій оригінального

конструктора Game Maker – Марк Овермарс, наступні версії, включаючи GameMaker: Studio (рис. 1), розробляються компанією YoYo Games. Безкоштовна версія (Standard) обмежена компіляцією під Windows. У порівнянні з нею, Professional версія має безліч переваг, включаючи управління ресурсами, компіляцію для macOS, Ubuntu і запуск на Android. Також, в професійній версії можна купувати окремі модулі, що розширюють функціональність програми. Версія Master Collection містить всі поточні модулі і майбутні доповнення версії 1.x.[5][1)]

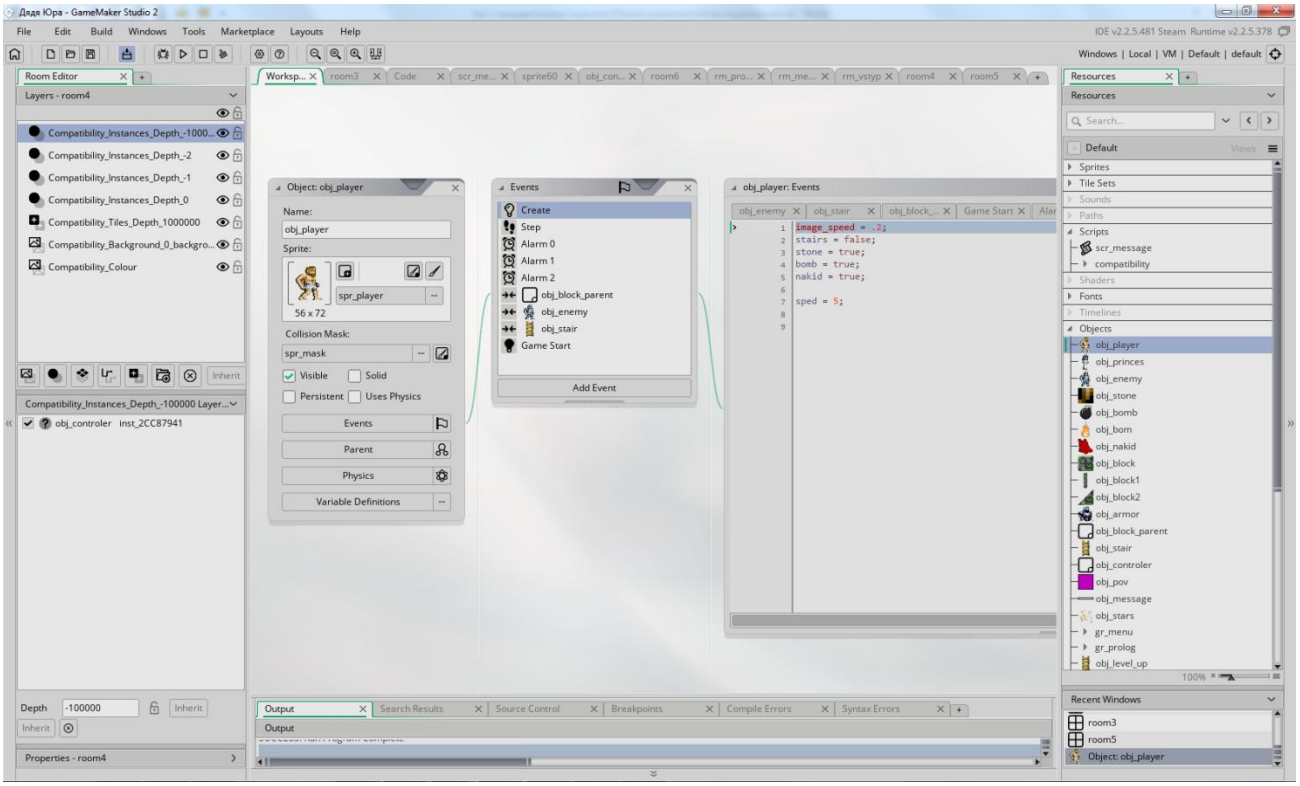

Рисунок 1− Вигляд IDE

2.2.2 Особливості ігрового движка GameMaker: Studio

GameMaker: Studio дозволяє писати розширення під безліч платформ на відповідних їм мовами. Підтримуються наступні типи файлів розширень: gml

---

[1)] [5] GameMaker Studio 2. URL: https://www.yoyogames.com (дата звернення 25.05.2020).

всюди, dll-бібліотеки на Windows, Windows Phone, Xbox One, js-скрипти для HTML5, so-бібліотеки на Linux і Tizen, dylib-бібліотеки на Mac, prx на PS4, suprs на PSVita, а також спеціальні placeholder для iOS і Android. Є можливість створити проксі-розширення для проектів на багатьох платформах, що дозволяє використовувати однакові назви функцій в коді, але звертатися до бібліотек відповідної платформи.

Є функції для роботи з кодуваннями Base64, JSON, MD5, SHA–1, можливості розпакування ZIP-архівів, читання і записи .ini, текстових і двійкових файлів, управління каталогами. Є можливість взаємодіяти з мережею: колективна гра по UDP, TCP, Bluetooth, відправка http-запитів, завантаження будь-яких файлів, взаємодія з Steam API і Facebook.

З версії 1.1.1086 додана підтримка шейдеров – ефективного інструменту управління отрисовкой, додавання графічних ефектів і перетворень. GameMaker: Studio підтримує вершинні і фрагменти шейдери на мовах GLSL ES, HLSL9, HLSL11 і GLSL.

З пристроїв, крім миші і клавіатури, присутні функції для взаємодії з джойстиками і геймпадом, для обробки торкань і нахилу на смартфонах.

GameMaker: Studio містить безліч математичних функцій для роботи зі скалярними і векторними величинами, включаючи тригонометричні обчислення, знаходження ступенів, логарифмів, інтерполяцій, нормалей векторів, скалярних творів. Є вбудований фізичний движок Box2D, набір функцій для роботи з ним, демонстраційні проекти Angry Cats і Angry Cats Space.

На відміну від GameMaker, Studio використовує компілятор, а не інтерпретатор коду, що прибрало можливість виконання чистого коду "на льоту", але істотно збільшило продуктивність ігор. Також є додатковий модуль YoYoCompiler, що транслює GML-код в C ++ і оптимізує його. Проекти, зібрані з використанням цього компонента, показують хорошу продуктивність з

сотнями ігрових елементів на рівні. Однак цей модуль доступний не для всіх платформ.

Є підтримка багатьох сервісів монетизації (таких як AdMob, Google Analytics, Google Play Licensing) з коду. Також є підтримка систем управління версіями – є приклади інтеграції з SVN, GIT, Mercurial. [6][1)]

В останніх версіях з'явилася безліч опцій налагодження програм в новому вікні The Debug Module, що дозволяють ставити крапки зупинки, стежити за виконанням коду, значеннями будь-яких змінних, конкретних об'єктів, вмістом буферів, кількістю і часом виконання різних скриптів.

2.2.3 Переваги й недоліки ігрового движка

Переваги:

– новий графічний редактор. Дуже сильно розробники поміняли графічний редактор. Не буду розповідати про кожен новий момент тому це зайняло б занадто багато часу і тексту. Інтерфейс став в рази краще. З'явилася можливість зручно малювати анімації прямо в самому двигуні не використовуючи сторонні програми;
– ігри стали працювати швидше Зізнаюся – сам не перевіряв. Однак почитавши форуми та відгуки я прийшов до висновку що YoYo поліпшили внутрішню оптимізацію через що ігри йдуть краще ніж в старій версії. Що в принципі – природно;
– зовнішній вигляд однозначно плюс. Інтерфейс виглядає красиво, сучасно і не перевантажено. Однак! Він – додаткова причина сжіранія нашого процесора з пам'яттю. Загалом палиця на два кінці. З одного боку ми маємо приємний візуал, з іншого – більш вимогливий до ресурсів движок;

[1)] [6] Jacob Habgood, Mark Overmars. The Game Maker's Apprentice: Game Development for Beginners.: Apress.: 2006. 314 p.

– актуальність. Розробники ведуть над ГМС2 роботу і поновлення регулярні. Останні апдейти дійсно гарні. Всі основні ресурси YoYo будуть швидше за все спрямовані на поліпшення GMS2 і його підтримку. Зараз їх головне завдання – змусити поміняти думку тих, хто дуже сильно незлюбив новий софт. Втім, це буде досить важко в силу того що люди дуже вперті;

– міжплатформовий це здатність створювати власні ігри для різних платформ для ПК (Windows, Mac OS, Ubuntu), мобільних (Android, Amazon Fire, iOS, Windows 10 Mobile), Web (HTML5) і Ігрових консолей (Xbox One, PlayStation 4);

– підтримка бібліотек і розширень, в тому числі на різних мовах;

– гнучка цінова категорія, Standard версія Game Maker: Studio абсолютно безкоштовна;

– власний спрощений мова програмування Game Maker Language (GML);

– інтеграція з декількома системами управління версіями;

– інтеграція з Steam, GooglePlay, AppStore;

Недоліки:

– GMS 2 вимагає більше ресурсів для роботи ніж старі версії. Старі GM у мене літали, ця вже гріє комп'ютер, що ні їсти приємно. Жере порівняно багато оперативки. Коли робиш гри місяцями і ще коротше інші програми фоном – це може напружувати. Ну і взагалі він працює повільніше, довше компіляція, довше завантаження і т.д. Це проблема була і у багатьох інших старих версій GM;

– російської версії поки немає, і може бути не буде ніколи. Хоча так-то гри робляться англійською, звикайте. Довідки на Російському теж нібито поки що немає, і ресурсів для навчання мало. Не знаю буду-чи я на своєму каналі робити про це навчальні відео, можливо – так, але в

цілому ГМС 2 сильно схожий на ранні версії і багато актуально і для нього;

– елементи інтерфейсу займають багато місця. Те, що одна вкладка не може насуватися на іншу, трохи дратує, особисто мене. Таким чином все займає більше місця і є відчуття що робили – аби красиво, а зручність це справа друга;
– незважаючи на можливість роботи з 3D, в Game Maker вона вкрай незручна;
– саме середовище розробки Game Maker: Studio доступна тільки на Windows (GMS 2 також доступна на macOS).

2.2.4 Аналіз Game Maker Language

Game Maker Language (GML) – це інтерпретована мова програмування, розроблений для використання разом з програмою для розробки комп'ютерних ігор званої Game Maker. Спочатку підтримка мови була впроваджена в Game Maker Марк Овермарс для доповнення системи кнопкових подій, проте пізніше все кнопкові події були включені в GML, дозволяючи програмісту уникнути використання кнопкових функцій. GML дуже сильно пов'язаний із середовищем Game Maker. Game Maker організований так, щоб не було необхідності програмування вручну таких речей, як управління подіями, дизайн рівнів і настройка об'єктів. Існує хибна думка, що GML підтримує вставки фрагментів коду на інших мовах, таких як Pascal, Assembler або C ++. Помилка виникла через часткову схожість синтаксису GML з Pascal і C ++.

Бібліотеки в Game Maker сукупність кнопкових подій утворює бібліотеку. В інтерфейсі програми бібліотеки відображаються як закладки, в яких знаходяться різні іконки подій. Кожне таке подія – це GML-скрипт або функція, яку користувач може використовувати в грі. В поставку Game Maker входять кілька стандартних бібліотек, які містять основні події, які використовуються в

більшості ігор. Також існує можливість створювати свої власні бібліотеки, використовуючи Library Maker. У GMS2 вбудований механізм конвертації дій, заданих «кнопками», в код GML і назад, що дозволяє новачкам швидше перейти на GML і покращує розуміння того, як працюють стандартні дії. [7][1)]

Розподіл пам'яті GML автоматично розподіляє пам'ять під змінні на ходу, і використовує динамічні типи, тому присвоювання змінним значення різних типів також можливо. Наприклад, спочатку можна створити цілочисельну змінну, а потім змінити її на строкову:

```
intNumber = 1;
intNumber = "Эта переменная теперь содержит строку";
```

У GML немає спеціального функціоналу, що дозволяє звільняти пам'ять, зайняту під змінну, однак при необхідності можна привласнювати змінної нове значення, меншого розміру. Наприклад, якщо у вас є змінна, в якій зберігається великий текст, то, присвоївши змінної значення порожнього рядка, можна домогтися вивільнення пам'яті. Те ж саме стосується і масивів:

```
data = [1, 2, 3, 4, 5]; // создали массив (такой синтаксис
создания массивов доступен начиная с GMS2)
data = 0; // уничтожили массив (теперь это просто переменная)
```

При знищенні об'єкта також знищуються всі змінні, локальні для нього, а будь-які глобальні змінні існують незалежно від них. Тому перевагу потрібно віддавати локальним змінним, а глобальні змінні використовувати тільки в разі реальної необхідності цього. Для зберігання великих обсягів інформації більш ефективно, в Game Maker є підтримка декількох структур даних – таких, як стек, чергу, список, карта, пріоритетна чергу і сітка. Ці структури створюються, модифікуються і знищуються за допомогою вбудованих функцій. Також є

[1)] [7] Matthew DeLucas. GameMaker Game Programming with GML.: Packt.: 2014. 350 p.

функції практично у всіх структурах для сортування даних в них. У деяких випадках більш зручним і ефективнішим буде використання буферів, що дозволяють зберігати довільні дані і є, по суті, просто виділеними шматками пам'яті.

Об'єкти і ресурси в основі роботи Game Maker з ресурсами лежать унікальні ідентифікатори, які служать для визначення конкретного ресурсу або примірника об'єкта. Ці ідентифікатори можуть бути використані скриптами або функціями для вказівки необхідного ресурсу. Так як створення ресурсів безпосередньо в Game Maker має на увазі вказівку імені, то це ім'я служить константою, що містить ідентифікатор ресурсу. Ідентифікатор конкретного екземпляра зберігається в локальній змінній «id». При динамічному створенні ресурсів, завжди повертається ідентифікатор створеного ресурсу, який може бути використаний в подальшому.

Ось простий приклад скрипта, який виводить легендарну фразу «Hello, World!»:

```
show_message("Hello, World!");
```

Ще один приклад, який відображає той же текст, але вже у вікні програми. Зауважте, що за умовчанням Game Maker безперервно перемальовує вікно, тому в звичайному випадку цей код потрібно помістити в подія малювання.

```
draw_text(10, 10, "Hello, World!");
```

Ось фрагмент коду з гри на GML:

```
var xx, yy, nn;
if can_shoot = true then begin
   can_shoot := false
   alarm[0] := 5
   xx := x + lengthdir_x(14, direction)
```

```
   yy := y + lengthdir_y(14, direction)
   nn := instance_create(xx, yy, obj_bullet)
   with nn begin
      speed := obj_tank.speed + 3
      direction := obj_tank.direction
   end
end
```

Ось приклад звичайного управління з клавіатури. Функція motion_set приймає два параметри: напрямок (в градусах) і швидкість (пікселів за крок). Виклик цієї функції призведе до зміни вбудованих локальних змінних speed і direction, які використовує Game Maker для переміщення об'єктів (об'єкти також можна зрушувати безпосередньо, використовуючи локальні змінні x і y):

```
if (keyboard_check(vk_left)) motion_set(180,4);
if (keyboard_check(vk_up)) motion_set(90,4);
if (keyboard_check(vk_right)) motion_set(0,4);
if (keyboard_check(vk_down)) motion_set(270,4);
if (keyboard_check(vk_nokey)) motion_set(0,0);
```

2.2.5 Аналоги Game maker studio

Unity – багатоплатформовий інструмент для розробки дво- та тривимірних додатків та ігор, що працює на операційних системах Windows і OS X. Створені за допомогою Unity застосунки працюють під системами Windows, OS X, Android, Apple iOS, Linux, а також на гральних консолях Wii, PlayStation 3 і XBox 360. Є можливість створювати інтернет-додатки за допомогою спеціального під'єднуваного модуля для браузера Unity, а також за допомогою експериментальної реалізації в межах модуля Adobe Flash Player. Застосунки, створені за допомогою Unity, підтримують DirectX та OpenGL.

Construct 2 – дозволяє створювати 2D-ігри різних жанрів і складності без навичок програмування . Ігри, зроблені на ньому, можуть бути доступні на PC, Mac, Linux, в браузерах з підтримкою HTML5, Android, iOS, Windows Phone, Blackberry 10, Amazon Appstore, Chrome Web Store, Facebook та ін. Підтримка

iOS і Android здійснюється завдяки технологіям CocoonJS від Ludei, directCanvas від appMobi і Intel XDK, які використовують апаратне прискорення для збільшення продуктивності HTML5 ігор в 5–10 разів. Інтерфейс програми має візуальний (WYSIWYG) редактор, можливо створити гру без навичок програмування. У редакторі є «події» (англ. Events) і «дії» (англ. Actions), що створюють логіку ігр. До додатком випускаються оновлення. Редактор позиціонує себе як «відповідний для людей з різним рівнем досвіду програмування». Доступна функція створення прототипу гри, демоверсії, презентацій, навчальних програм. За даними на 24 березня 2017 року велась розробка програми Construct 3.

Godot – повнофункціональний, з відкритим вихідним кодом, MIT ліцензований ігровий движок. Він фокусується на відмінних інструментах і візуально орієнтованому робочому процесі, який може експортуватися на ПК, мобільні і веб-платформи. Це крос-платформний 2D і 3D ігровий движок і розробка ігор IDE. Ви можете переносити свою гру в Windows, Linux, Mac OSx / iOS, BSD, Android, BlackBerry 10, HTML5, PlayStation 3, PlayStation Vita і Nintendo 3DS. Особливості Візуальний редактор, повний зрілих інструментів. Недосліджений призначений для користувача інтерфейс з контекстно-залежними редакторами. Відмінна система сцен, підтримка instancing і inheritance. Дружні інтерфейси створення контенту для художників, дизайнерів рівнів, аніматорів. Візуальне редагування шейдеров для 2D і 3D

Unreal Engine – це професійний набір інструментів і технологій, що використовуються для створення високоякісних ігор на різних платформах. Архітектура рендеринга Unreal Engine дозволяє розробникам створювати приголомшливі візуальні ефекти, а також елегантно розподіляти по більш низьких рівнів. Революційні функції нового робочого процесу і глибокий набір інструментів дозволяють розробникам швидко переходити по ідеям і бачити негайні результати, в той час як повний доступ до вихідного коду на C ++ дає

досвід абсолютно новий рівень. Технологія Unreal Engine забезпечує сотні ігор, а також 3D-фільми в реальному часі, навчальні симуляції, візуалізацію і багато іншого. За останні 15 років тисячі людей і команд і побудували кар'єри і компанії по навичкам, розробленим з використанням двигуна.

Cocos2d – крос-платформний фреймворк, який використовується для розробки інтерактивних додатків і ігор (переважно для мобільних пристроїв). Є відкритим програмним забезпеченням. Cocos2d містить безліч відгалужень, таких як Cocos2d-ObjC, Cocos2d-x, Cocos2d-html5 і Cocos2d-XNA. Також в співтоваристві Cocos2d є кілька незалежних редакторів, призначених для редагування спрайтів, частинок, шрифтів і тайлових карт. Можна також згадати редактори світу: CocosBuilder і CocoStudio.

3 РЕАЛІЗАЦІЯ ТЕХНІЧНОГО ПРОЕКТУ

3.1 Розробка програмної системи

Метод пересування героя та його стан покою:

```
//Движение
    if keyboard_check(ord("D")) && place_free(x + 5, y) &&
!keyboard_check(ord("A")) {
        if place_free(x, y + 1) {
            if place_free(x + 5, y + vspeed) {
                x += sped;
                sprite_index = spr_playergo;
                image_xscale = 1;
                image_speed = .2;
            }
        } else {
            x += sped;
            sprite_index = spr_playergo;
            image_xscale = 1;
            image_speed = .2;
        }
    } else {
        if keyboard_check(ord("A")) && place_free(x - 5, y) &&
!keyboard_check(ord("D")) {
            if place_free(x, y + 1) {
                if place_free(x - 5, y + vspeed) {
                    x -= sped;
                    sprite_index = spr_playergo;
                    image_xscale = -1;
                    image_speed = .2;
                }
            } else {
                x -= sped;
                sprite_index = spr_playergo;
                image_xscale = -1;
                image_speed = .2;
            }
        } else {
            sprite_index = spr_player;
            image_speed = .1;
```

Метод вибору зброї головного героя:

```
            //Выбор оружия
            if keyboard_check(ord("1")) {
                obj_controler.weapon = 1;
            } else {
                if keyboard_check(ord("2")) {
```

```
                    obj_controler.weapon = 2;
                } else {
                    if keyboard_check(ord("3")) {
                        obj_controler.weapon = 3;
                    }
                }
            }
```

Метод роботи різної зброї головного героя:

```
            //Бросок камня 1
            if keyboard_check(vk_space) && gravity = 0 && stone =
true && obj_controler.weapon = 1 && obj_controler.stones > 0 {
                alarm[0] = 25;
                stone = false;
            } else {
                if alarm[0] > 11 && alarm[0] < 20 {
                    sprite_index = spr_player_stone;
                    image_speed = 0;
                    image_index = 0;
                } else {
                    if alarm[0] > 1 && alarm[0] < 10 {
                        sprite_index = spr_player_stone;
                        image_speed = 0;
                        image_index = 1;
                    } else {
                        if obj_controler.stones = 0 &&
keyboard_check_pressed(vk_space) && obj_controler.weapon = 1 {

                        }
                    }
                }
            }
            //Установка бомбы 2
            if keyboard_check(vk_space) && gravity = 0 && bomb =
true && obj_controler.weapon = 2 && obj_controler.mines > 0 {
                instance_create(x, y, obj_bomb);
                alarm[1] = 50;
                bomb = false;
                obj_controler.mines -= 1;
            }
            //Одевание плаща 3
            if keyboard_check(vk_space) && gravity = 0 && nakid =
true && obj_controler.weapon = 3 && obj_controler.nakids > 0 {
                instance_create(x, y, obj_nakid);
                alarm[2] = 60;
                nakid = false;
                obj_controler.nakids -= 1;
            } else {
                if nakid = true {
                    alarm[2] = 0;
                }
            }
        }
    }
```

Методи диалогової системи, переходу на наступний рівень та повідомлення про закінчення відеогри:

```
//Диалоговые окна
scr_message();

//Переход на следующий уровень
if place_meeting(x, y, obj_level_up) {
    room_goto_next();
}
if room = room5 {
    show_message("Ждите новых обновлений.")
    room_goto(rm_menu)
    with obj_controler {
        instance_destroy();
    }
}
```

На рис. 2 показано роботу діалогової системи.

```
1  if place_meeting(x,y,obj_stop_parent){
2      sped = 0;
3      with obj_message {visible = true;}
4      //Нахождение камня
5      if place_meeting(x,y,obj_stop_stone){
6          with obj_message{
7              with obj_stop_stone {instance_destroy();}
8              mytext = ("У меня есть несколько камней. # Надеюсь мне их хватит.#(Чтобы использовать предмет - нажмите \"пробел\")");
9              alarm[0] = 60;
10         }
11     }
12     //Нахождение бомбы
13     if place_meeting(x,y,obj_stop_bomb){
14         with obj_message{
15             with obj_stop_bomb{instance_destroy();}
16             mytext = ("Думаю, что бомбы лишними не будут, возьму с собой парочку.")
17             alarm[0] = 45;
18             obj_controler.mines = 3;
19             obj_controler.mines_w = 1;
20         }
21     }
22     //Тупик
23     if place_meeting(x,y,obj_stop_typ){
24         with obj_message{
25             with obj_stop_typ {instance_destroy();}
26             mytext = ("Похоже на тупик. Надо найти другой путь.")
27             alarm[0] = 45;
28         }
29     }
30     //Стражники
31     if place_meeting(x,y,obj_stop_strag){
32         with obj_message{
33             with obj_stop_strag {instance_destroy();}
34             mytext = ("А вот и стражники. Надо попробывать их обойти.")
35             alarm[0] = 45;
36         }
37     }
38     //Стена
39     if place_meeting(x,y,obj_stop_bom){
40         with obj_message{
41             with obj_stop_bom{instance_destroy();}
42             mytext = ("Пора испробывать эти бомбы в деле. #Похоже, что это ветхая стена на неё и испробуем.")
43             alarm[0] = 60;
44         }
45     }
```

Рисунок 2 – Код роботи діалогової системи

Методи гравітації та стрибку головного герою:

```
//Гравитация
if (place_free(x, y + 1)) {
    gravity_direction = 270;
    gravity = .75;
    sprite_index = spr_player_jump;
    if vspeed >= 0 {
        image_index = 0;
    } else {
        image_index = 1;
    }

    if vspeed >= 13 {
        vspeed = 13;
    }
} else {
    gravity = 0

    //Прыжок
    if keyboard_check(ord("W")) && !place_free(x, y + 1) {
        vspeed -= 13;
    }
}
```

Методи роботи штучного інтелекту та його пересування:

```
//dir = 1 - вправо; dir = 2 - влево; dir = 3 - бежит на игрока;
dvig = 1 - стоит; dvig = 0 - идет;
 if dir = 1 && place_free(x+4,y) && dvig = 0{
    sprite_index = spr_enemy_go;
    image_xscale = 1;
    image_speed = .2;
    sped = 4;
    x += sped;
    if !place_free(x+4,y) && dvig = 0{
        alarm[0] = 60
        sprite_index = spr_enemy;
        image_xscale = 1;
        image_speed = .1;
        dvig = 1;
    }
 } else {
    if dir = 2 && place_free(x-4,y) && dvig = 0{
        sprite_index = spr_enemy_go;
        image_xscale = -1;
        image_speed = .2;
        sped = 4;
        x -= sped;
        if !place_free(x-4,y) && dvig = 0{
            alarm[0] = 60
            sprite_index = spr_enemy;
            image_xscale = -1;
            image_speed = .1;
```

```
        dvig = 1;
    }
```

Метод виявлення гравця штучним інтелектом та пересування до нього:

```
    if point_distance(x,y,obj_player.x,obj_player.y) < 150 && (dir
= 1 && obj_player.x>x) or (dir = 2 && obj_player.x<x){
    if dir = 1 && place_free(x+4,y) && dvig = 0{
         sprite_index = spr_enemy_go;
          image_xscale = 1;
          image_speed = .2;
          sped = 5;
          x += sped;
          if !place_free(x+8,y) && place_free(x,y-1){
                   if place_free(x+5,y+vspeed){
                        x += sped;
                      sprite_index = spr_enemy_go;
                      image_xscale = 1;
                      image_speed = .2;
                 }
                 if !place_free(x,y+1){
                      vspeed -= 10;
                 }
          }
    } else {
          if dir = 2 && place_free(x-4,y) && dvig = 0{
                  sprite_index = spr_enemy_go;
                  image_xscale = -1;
                  image_speed = .2;
                  sped = 5;
                  x -= sped;
                  if !place_free(x-8,y) && place_free(x,y-1){
                       if place_free(x-5,y+vspeed){
                        x -= sped;
                            sprite_index = spr_enemy_go;
                            image_xscale = -1;
                            image_speed = .2;
                        }
                        if !place_free(x,y+1){
                            vspeed -= 10;
                        }
```

На рис. 3 зображено код об’єкту, який контролює деякі дії які проходять на екрані.

```
draw_set_colour(c_red)
if stone_w = 1{
    draw_sprite(spr_stone_s,1,__view_get( e__VW.XView, 0 )+24,__view_get( e__VW.YView, 0 )+34)
        draw_text(__view_get( e__VW.XView, 0 )+20,__view_get( e__VW.YView, 0 )+62,string_hash_to_newline(stones))
        if mines_w = 1{
            draw_sprite(spr_bomb_s,1,__view_get( e__VW.XView, 0 )+76,__view_get( e__VW.YView, 0 )+32)
            draw_text(__view_get( e__VW.XView, 0 )+70,__view_get( e__VW.YView, 0 )+62,string_hash_to_newline(mines))
            if nakid_w = 1{
                draw_sprite(spr_nakid_s,1,__view_get( e__VW.XView, 0 )+126,__view_get( e__VW.YView, 0 )+34)
                draw_text(__view_get( e__VW.XView, 0 )+120,__view_get( e__VW.YView, 0 )+62,string_hash_to_newline(nakids))
            }
        }
    }
    if weapon = 1{
        draw_button(__view_get( e__VW.XView, 0 )+1,__view_get( e__VW.YView, 0 )+77,__view_get( e__VW.XView, 0 )+49,__view_get( e__VW.YView, 0 )+89,0)
    } else {
        if weapon = 2{
            draw_button(__view_get( e__VW.XView, 0 )+51,__view_get( e__VW.YView, 0 )+77,__view_get( e__VW.XView, 0 )+100,__view_get( e__VW.YView, 0 )+89,0
        } else {
            if weapon = 3{
                draw_button(__view_get( e__VW.XView, 0 )+102,__view_get( e__VW.YView, 0 )+77,__view_get( e__VW.XView, 0 )+151,__view_get( e__VW.YView, 0 )
            }
        }
    }
    if hp > 2{
        draw_sprite(spr_life,1,__view_get( e__VW.XView, 0 )+22,__view_get( e__VW.YView, 0 )+112);
        draw_sprite(spr_life,1,__view_get( e__VW.XView, 0 )+62,__view_get( e__VW.YView, 0 )+112);
        draw_sprite(spr_life,1,__view_get( e__VW.XView, 0 )+102,__view_get( e__VW.YView, 0 )+112);
    } else {
        if hp > 1{
            draw_sprite(spr_life,1,__view_get( e__VW.XView, 0 )+22,__view_get( e__VW.YView, 0 )+112);
            draw_sprite(spr_life,1,__view_get( e__VW.XView, 0 )+62,__view_get( e__VW.YView, 0 )+112);
        } else {
            if hp > 0{
                draw_sprite(spr_life,1,__view_get( e__VW.XView, 0 )+22,__view_get( e__VW.YView, 0 )+112);
            }
        }
    }
```

Рисунок 3 – Код об’єкту контролера відеогри

3.2 Опис геймплея

Відеогра починається із знайомства з головним героєм, якого звуть «Дядя Юра». Саме він нас зустрічає на предпоказі до початку гри (рис. 4).

Рисунок 4 – Заставка перед початком гри

Герой зі своєю коханою потрапляє в халепу, по сюжету гри її викрадають. Після, за більшістю сюжетів гри, йому належить її врятувати, проходячи по всіх рівнях замку і всіляко уникаючи або усуваючи ворогів. В одному з рівнів гри він все-таки знаходить свою кохану, але зненацька з'являється демон, який знов викрадає її, але вже в пекло, після чого нашому головному героєві знов належить подорож, але вже трохи в іншому напрямку – в пекло.

Після цього потрапляємо до меню (рис. 5) де можна обрати «Начать спасение», подивитися авторів чи просто вийти з додатку.

Рисунок 5 − Меню

При обрані «Начать спасение», починається перший рівень (рис. 6), на якому гравець знайомиться з управлінням персонажу, а також присутній опис про геймплейну особистість «кидання каміння». У верхньому куті зліва

присутня серця – вони показують скільки залишилося життів у героя, та скільки каменів. Якщо герой помирає, то в нього пропадає одне життя та рівень починається спочатку, якщо гравець втратить всі життя, то гра закінчиться, та поверне нас у головне меню.

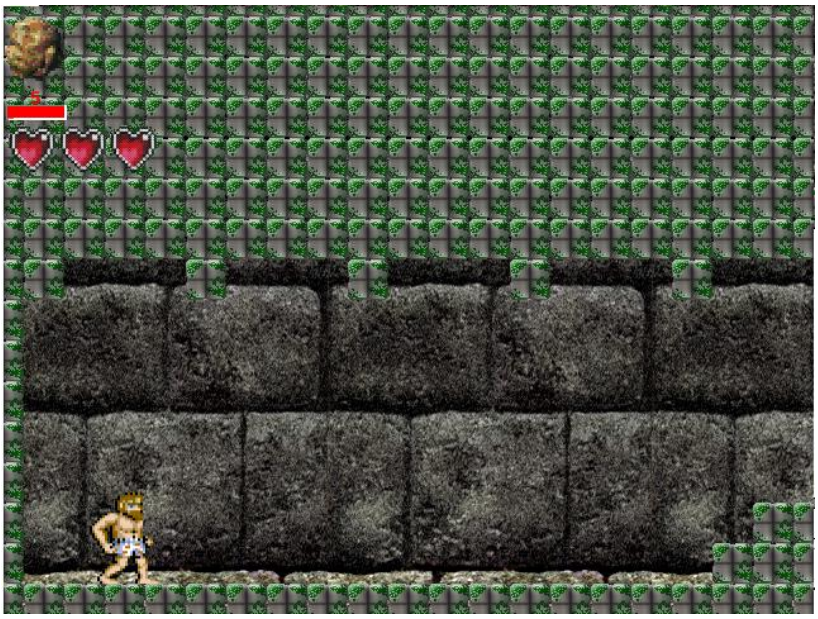

Рисунок 6 − Початок першого рівня

Ще декілька рівнів ми можемо побачити на рис. 7 та 8:

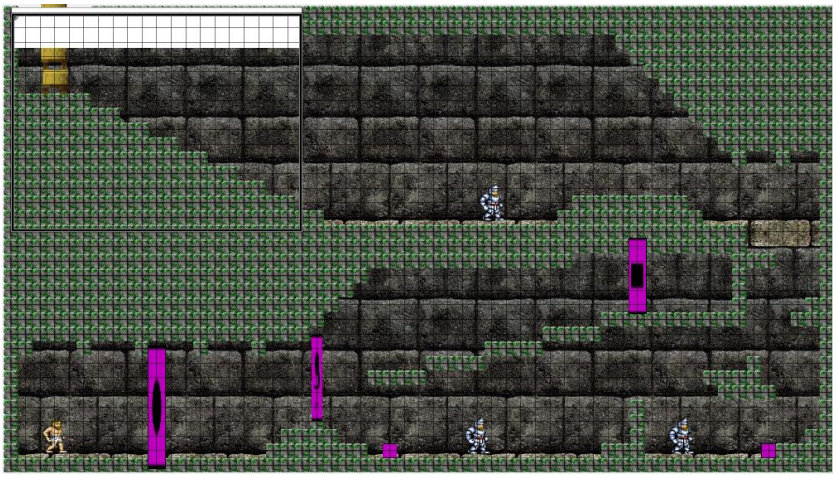

Рисунок 7 − Повний перший рівень

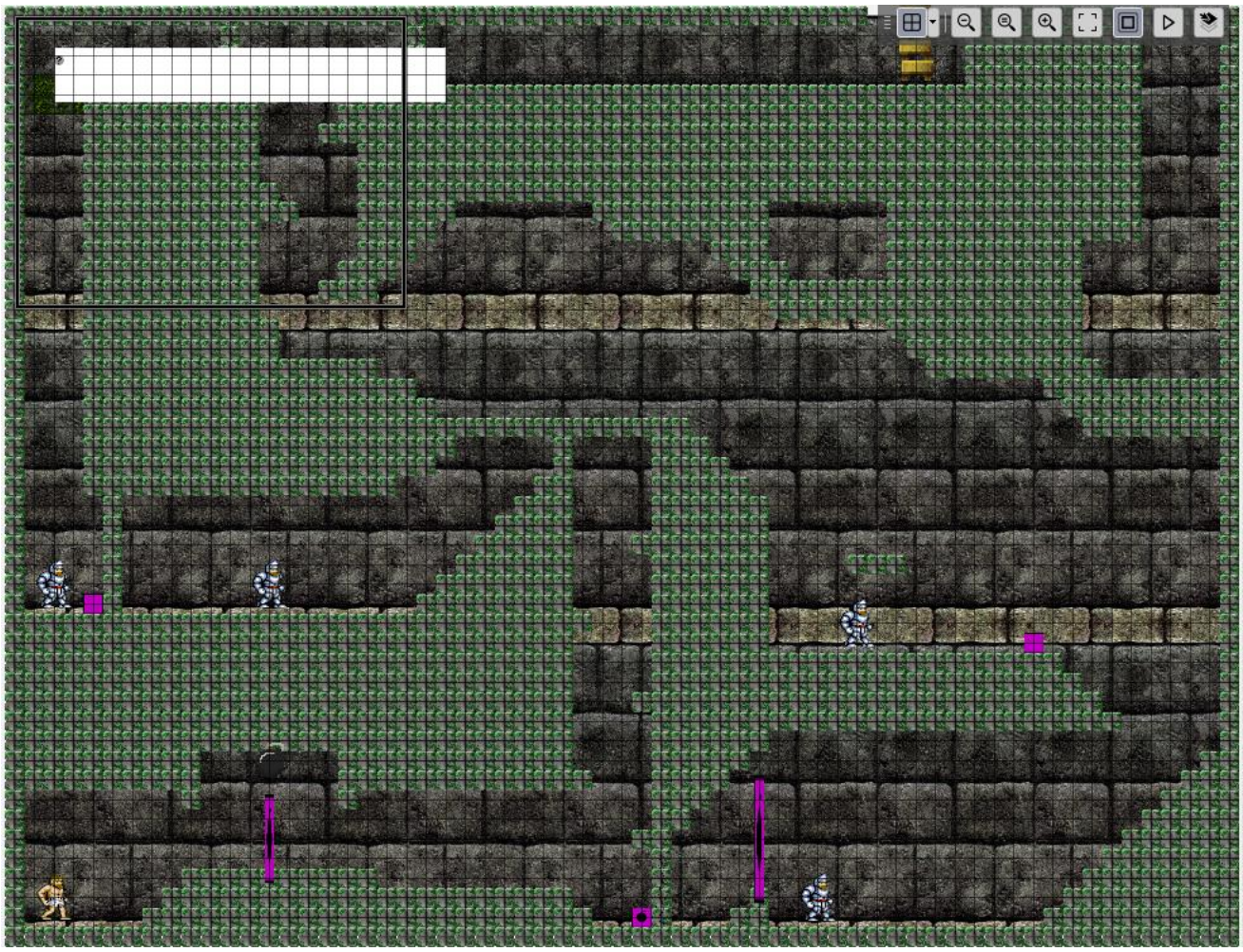

Рисунок 8 − Повний другий рівень

3.3 Опис сюжету та персонажів

Головний герой з дружиною, які живуть своїм життям неподалік від невеликого містечка. Лицар Комбриг за сюжетом гри шукає собі принцесу. Йому дуже сподобалася дружина нашого головного героя. Він вирішує вкрасти її, не знаючи про її чоловіка, який готовий пожертвувати життям заради неї.

Але це лише перша частина сюжету.

Друга полягає в тому – що після того, як Демон краде принцесу у Комбрега прямо перед очима головного героя, то Комбриг вирішує допомогти, даючи декілька завдань, після яких отримуємо для порятунки та проходження необхідні ресурси. У той же час можна відмовитися від них та піти відразу на

спасіння, але зайвими ці ресурси не виявляться, в той же час вони не впливають на «непрохідність» гри. Після того, як ми пройдемо всі 8 кіл пекла, на 9-ому зустрічається з самим демон і він повинен здолати його.

Персонажі:

Головний герой (рис. 9) – гравець управляє безпосередньо всіма діями головного героя і впливає на те, врятує наш головний герой принцесу чи ні.

Рисунок 9 − Головний герой

Лицар Комбриг (рис. 10) – перший лиходій (який в дійсності допомагає герою) − просто людина, яка вирішила зробити необдуману дію.

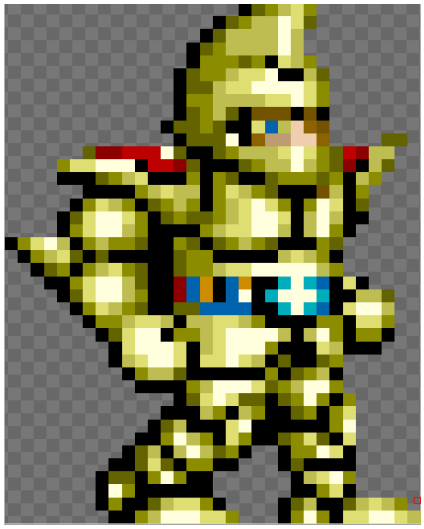

Рисунок 10 − Лицар Комбриг

Демон – головний лиходій – це вже не той Комбриг, який не мав жодних поганих намірів у крадіжці принцеси, це істота яка вкрала її, щоб наш головний герой більше ніколи не побачився зі своєю коханою.

Принцеса (рис. 11) – сама мовчазна і сама малозадіяна особистість в грі, проте весь сюжет гри крутиться навколо цієї особистості.

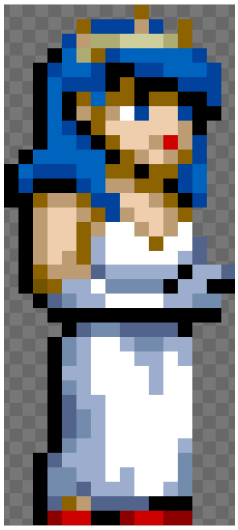

Рисунок 11 – Принцеса

ВИСНОВКИ

В процесі розробки дипломного проекту були реалізовані усі поставленні завдання та отримано готове програмне забезпечення з простим та інтуїтивно-зрозумілим користувальницьким інтерфейсом, що задовольняє основним сучасним вимогам щодо призначеного для користувача ПО.

Була розроблена логіка програми, також для користувачів було розроблено зручний інтерфейс програми, для користування без зайвих перешкод.

Програмний модуль повністю виконує ті функції, які описані в постановці завдання. Після розробки програмного модуля була створена документація для користувачів і адміністраторів системи.

Функціонал програми відповідає раніше поставленим вимогам щодо необхідних можливостей програмного продукту.

Були проведені різноманітні тести на виявлення різних помилок програми, і при виявленні таких усувалися та повторно перевірялися.

Після розробки, програму можна вдосконалювати увесь час, робити її більш функціональною та більш зручною для користування.

ПЕРЕЛІК ПОСИЛАНЬ

1. Йоган Гейзинга. Homo Ludens. Досвід визначення ігрового елемента культури.: Издательство Ивана Лимбаха.: 2016. 416 с.
2. Wolf M. J. P. Encyclopedia of Video Games: The Culture, Technology, and Art of Gaming Santa Barbara CA: Greenwood.: 2012. 740 p.
3. Розробка гри на JavaScript. URL: https://webref.ru/mobile/firefoxosgame/levels (дата звернення 23.05.2020).
4. С. Шапкин. Компьютерная игра: новая область психологических исследований.: "Психологический журнал": 1999. том 20. № 1. С. 86–102.
5. GameMaker Studio 2.. URL: https://www.yoyogames.com (дата зверенння 25.05.2020).
6. Jacob Habgood, Mark Overmars. The Game Maker's Apprentice: Game Development for Beginners.: Apress.: 2006. 314 p.
7. Matthew DeLucas. GameMaker Game Programming with GML.: Packt.: 2014. 350 p.

ДОДАТОК

Лістинг програми

```
    //Движение
    if keyboard_check(ord("D")) && place_free(x + 5, y) &&
!keyboard_check(ord("A")) {
        if place_free(x, y + 1) {
            if place_free(x + 5, y + vspeed) {
                x += sped;
                sprite_index = spr_playergo;
                image_xscale = 1;
                image_speed = .2;
            }
        } else {
            x += sped;
            sprite_index = spr_playergo;
            image_xscale = 1;
            image_speed = .2;
        }
    } else {
        if keyboard_check(ord("A")) && place_free(x - 5, y) &&
!keyboard_check(ord("D")) {
            if place_free(x, y + 1) {
                if place_free(x - 5, y + vspeed) {
                    x -= sped;
                    sprite_index = spr_playergo;
                    image_xscale = -1;
                    image_speed = .2;
                }
            } else {
                x -= sped;
                sprite_index = spr_playergo;
                image_xscale = -1;
                image_speed = .2;
            }
        } else {
            sprite_index = spr_player;
            image_speed = .1;

            //Выбор оружия
            if keyboard_check(ord("1")) {
                obj_controler.weapon = 1;
            } else {
                if keyboard_check(ord("2")) {
                    obj_controler.weapon = 2;
                } else {
                    if keyboard_check(ord("3")) {
                        obj_controler.weapon = 3;
                    }
                }
            }

            //Бросок камня 1
            if keyboard_check(vk_space) && gravity = 0 && stone = true &&
obj_controler.weapon = 1 && obj_controler.stones > 0 {
                alarm[0] = 25;
                stone = false;
            } else {
                if alarm[0] > 11 && alarm[0] < 20 {
                    sprite_index = spr_player_stone;
                    image_speed = 0;
```

```
                    image_index = 0;
                } else {
                    if alarm[0] > 1 && alarm[0] < 10 {
                        sprite_index = spr_player_stone;
                        image_speed = 0;
                        image_index = 1;
                    } else {
                        if obj_controler.stones = 0 &&
keyboard_check_pressed(vk_space) && obj_controler.weapon = 1 {

                        }
                    }
                }
            }
            //Установка бомбы 2
            if keyboard_check(vk_space) && gravity = 0 && bomb = true &&
obj_controler.weapon = 2 && obj_controler.mines > 0 {
                instance_create(x, y, obj_bomb);
                alarm[1] = 50;
                bomb = false;
                obj_controler.mines -= 1;
            }
            //Одевание плаща 3
            if keyboard_check(vk_space) && gravity = 0 && nakid = true &&
obj_controler.weapon = 3 && obj_controler.nakids > 0 {
                instance_create(x, y, obj_nakid);
                alarm[2] = 60;
                nakid = false;
                obj_controler.nakids -= 1;
            } else {
                if nakid = true {
                    alarm[2] = 0;
                }
            }
        }
    }

    //Диалоговые окна
    scr_message();

    //Переход на следующий уровень
    if place_meeting(x, y, obj_level_up) {
        room_goto_next();
    }
    if room = room5 {
        show_message("Ждите новых обновлений.")
        room_goto(rm_menu)
        with obj_controler {
            instance_destroy();
        }
    }


    //Гравитация
    if (place_free(x, y + 1)) {
        gravity_direction = 270;
        gravity = .75;
        sprite_index = spr_player_jump;
        if vspeed >= 0 {
            image_index = 0;
        } else {
            image_index = 1;
        }

        if vspeed >= 13 {
            vspeed = 13;
        }
```

```
    } else {
        gravity = 0

        //Прыжок
        if keyboard_check(ord("W")) && !place_free(x, y + 1) {
            vspeed -= 13;
        }
    }
    strag = instance_nearest(x, y, obj_enemy)
}
if place_meeting(x,y,obj_stop_parent){
    sped = 0;
    with obj_message {visible = true;}
    //Нахождение камня
    if place_meeting(x,y,obj_stop_stone){
        with obj_message{
            with obj_stop_stone {instance_destroy();}
            mytext = ("У меня есть несколько камней. # Надеюсь мне их
хватит.#(Чтобы использовать предмет - нажмите \"пробел\")");
            alarm[0] = 60;
        }
    }
    //Нахождение бомбы
    if place_meeting(x,y,obj_stop_bomb){
        with obj_message{
            with obj_stop_bomb{instance_destroy();}
            mytext = ("Думаю, что бомбы лишними не будут, возьму с собой
парочку.")
            alarm[0] = 45;
            obj_controler.mines = 3;
            obj_controler.mines_w = 1;
        }
    }
    //Тупик
    if place_meeting(x,y,obj_stop_typ){
        with obj_message{
            with obj_stop_typ {instance_destroy();}
            mytext = ("Похоже на тупик. Надо найти другой путь.")
            alarm[0] = 45;
        }
    }
    //Стражники
    if place_meeting(x,y,obj_stop_strag){
        with obj_message{
            with obj_stop_strag {instance_destroy();}
            mytext = ("А вот и стражники. Надо попробывать их обойти.")
            alarm[0] = 45;
        }
    }
    //Стена
    if place_meeting(x,y,obj_stop_bom){
        with obj_message{
            with obj_stop_bom{instance_destroy();}
            mytext = ("Пора испробывать эти бомбы в деле. #Похоже, что это
ветхая стена на неё и испробуем.")
            alarm[0] = 60;
        }
    }
    //Взрыв стражников
    if place_meeting(x,y,obj_stop_bom_strag){
        with obj_message{
            with obj_stop_bom_strag{instance_destroy();}
            mytext = ("Интересно как понравится стражникам мой подарок из
пороха.")
            alarm[0] = 45;
        }
    }
```

```
}
draw_set_colour(c_red)
if stone_w = 1{
      draw_sprite(spr_stone_s,1,__view_get( e__VW.XView, 0 )+24,__view_get(
e__VW.YView, 0 )+34)
        draw_text(__view_get( e__VW.XView, 0 )+20,__view_get( e__VW.YView, 0
)+62,string_hash_to_newline(stones))
        if mines_w = 1{
            draw_sprite(spr_bomb_s,1,__view_get( e__VW.XView, 0 )+76,__view_get(
e__VW.YView, 0 )+32)
            draw_text(__view_get( e__VW.XView, 0 )+70,__view_get( e__VW.YView, 0
)+62,string_hash_to_newline(mines))
            if nakid_w = 1{
                draw_sprite(spr_nakid_s,1,__view_get( e__VW.XView, 0
)+126,__view_get( e__VW.YView, 0 )+34)
                draw_text(__view_get( e__VW.XView, 0 )+120,__view_get(
e__VW.YView, 0 )+62,string_hash_to_newline(nakids))
            }
        }
    }
    if weapon = 1{
        draw_button(__view_get( e__VW.XView, 0 )+1,__view_get( e__VW.YView, 0
)+77,__view_get( e__VW.XView, 0 )+49,__view_get( e__VW.YView, 0 )+89,0)
    } else {
        if weapon = 2{
            draw_button(__view_get( e__VW.XView, 0 )+51,__view_get( e__VW.YView,
0 )+77,__view_get( e__VW.XView, 0 )+100,__view_get( e__VW.YView, 0 )+89,0)
        } else {
            if weapon = 3{
                draw_button(__view_get( e__VW.XView, 0 )+102,__view_get(
e__VW.YView, 0 )+77,__view_get( e__VW.XView, 0 )+151,__view_get( e__VW.YView, 0
)+89,0)
            }
        }
    }
    if hp > 2{
        draw_sprite(spr_life,1,__view_get( e__VW.XView, 0 )+22,__view_get(
e__VW.YView, 0 )+112);
        draw_sprite(spr_life,1,__view_get( e__VW.XView, 0 )+62,__view_get(
e__VW.YView, 0 )+112);
        draw_sprite(spr_life,1,__view_get( e__VW.XView, 0 )+102,__view_get(
e__VW.YView, 0 )+112);
    } else {
        if hp > 1{
            draw_sprite(spr_life,1,__view_get( e__VW.XView, 0 )+22,__view_get(
e__VW.YView, 0 )+112);
            draw_sprite(spr_life,1,__view_get( e__VW.XView, 0 )+62,__view_get(
e__VW.YView, 0 )+112);
        } else {
            if hp > 0{
                draw_sprite(spr_life,1,__view_get( e__VW.XView, 0
)+22,__view_get( e__VW.YView, 0 )+112);
            }
        }
    }
//dir = 1 - вправо; dir = 2 - влево; dir = 3 - бежит на игрока; dvig = 1 -
стоит; dvig = 0 - идет;
 if dir = 1 && place_free(x+4,y) && dvig = 0{
    sprite_index = spr_enemy_go;
    image_xscale = 1;
    image_speed = .2;
    sped = 4;
    x += sped;
    if !place_free(x+4,y) && dvig = 0{
        alarm[0] = 60
        sprite_index = spr_enemy;
        image_xscale = 1;
```

```
        image_speed = .1;
        dvig = 1;
    }
} else {
    if dir = 2 && place_free(x-4,y) && dvig = 0{
        sprite_index = spr_enemy_go;
        image_xscale = -1;
        image_speed = .2;
        sped = 4;
        x -= sped;
        if !place_free(x-4,y) && dvig = 0{
            alarm[0] = 60
            sprite_index = spr_enemy;
            image_xscale = -1;
            image_speed = .1;
            dvig = 1;
        }
    } else {
        if point_distance(x,y,obj_player.x,obj_player.y) < 150 && (dir = 1 &&
obj_player.x>x) or (dir = 2 && obj_player.x<x){
            if dir = 1 && place_free(x+4,y) && dvig = 0{
                sprite_index = spr_enemy_go;
                image_xscale = 1;
                image_speed = .2;
                sped = 5;
                x += sped;
                if !place_free(x+8,y) && place_free(x,y-1){
                    if place_free(x+5,y+vspeed){
                        x += sped;
                        sprite_index = spr_enemy_go;
                        image_xscale = 1;
                        image_speed = .2;
                    }
                    if !place_free(x,y+1){
                        vspeed -= 10;
                    }
                }
            } else {
                if dir = 2 && place_free(x-4,y) && dvig = 0{
                    sprite_index = spr_enemy_go;
                    image_xscale = -1;
                    image_speed = .2;
                    sped = 5;
                    x -= sped;
                    if !place_free(x-8,y) && place_free(x,y-1){
                        if place_free(x-5,y+vspeed){
                            x -= sped;
                            sprite_index = spr_enemy_go;
                            image_xscale = -1;
                            image_speed = .2;
                        }
                        if !place_free(x,y+1){
                            vspeed -= 10;
                        }
                    }
                }
            }
        }
    }
}
//Столкновение с невидимым блоком
if place_meeting(x,y,obj_pov) && dir = 1 && dvig = 0{
    alarm[2] = 60;
    sprite_index = spr_enemy;
    image_xscale = 1;
    image_speed = .1;
    dvig = 1;
```

```
} else {
   if place_meeting(x,y,obj_pov) && dir = 2 && dvig = 0{
       alarm[2] = 60;
       sprite_index = spr_enemy;
       image_xscale = -1;
       image_speed = .1;
       dvig = 1;
   }
}

//Оглушение
if place_meeting(x,y,obj_stone){
   sprite_index = spr_enemy;
   image_speed = .1;
   dir = 0;
   alarm[1] = 60;
   if image_xscale = 1{
       instance_create(x+6,y-46,obj_stars)
   } else {
       instance_create(x-4,y-46,obj_stars)
   }
}

//Уничтожение с помощью бомбы
if place_meeting(x,y,obj_bom) && distance_to_object(obj_bomb <= 45){
   instance_change(obj_armor,1);
}


//Гравитация
if (place_free(x,y+1)){
   gravity_direction=270;
   gravity=.75;
   sprite_index = spr_enemy;
   if vspeed >= 0 {
       image_index = 0;
   } else {
       image_index = 1;
   }

   if vspeed >= 10 {
       vspeed = 10;
   }
} else {
   gravity = 0;
}
```