Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРССТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОННИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: программирование

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

на тему

**РАЗРАБОТКА ИГРЫ DOODLE JUMP**

Студент гр. 053501 К. Д. Криштафович

Руководитель: И. А. Удовин

Минск 2021

*Белорусский государственный университет информатики и радиоэлектроники*

*Факультет компьютерных систем и сетей*

«УТВЕРЖДАЮ»

Заведующий кафедрой информатики

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Н. А. Волорова

«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021 года

**ЗАДАНИЕ**

по курсовой работе

*студенту Криштафович К. Д.*

1. Тема курсовой работы «Разработка игры Doodle Jump».
2. Дата защиты курсовой работы «\_\_» \_\_\_\_\_\_\_\_\_\_ 2021 г.
3. Исходные данные для курсовой работы
   1. Язык программирования C++. Фреймворк Qt.
4. Содержание пояснительной записки
   1. Введение.
   2. Анализ предметной области.
      1. Предметная область.
      2. Обзор аналогичных проектов.
      3. Выбор языка и среды программирования.
      4. Постановка задачи.
   3. Описание программы.
      1. Общие сведения.
      2. Процесс разработки пользовательского интерфейса
      3. Игровой процесс.
         1. Структура классов.
         2. Логика взаимодействия объектов.
         3. Отслеживание движения объектов и действий игрока.
         4. Окончание игры.
         5. Музыкальное сопровождение.
   4. Список используемых источников.
5. Консультант по курсовой работе Удовин И. А.
6. Дата выдачи задания «\_\_\_» \_\_\_\_\_\_\_\_\_ 2021 г.
7. Календарный график выполнения курсовой работы.
   1. разделы 1 к 10.03.2021 г. – 30%
   2. разделы 2 к 15.04.2021 г. – 70%
   3. оформление кода программы к 20.05.2021 г. – 30%

Руководитель курсовой работы *И. А. Удовин*

Задание принял для исполнения *К. Д. Криштафович*

(дата и подпись студента)

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc73918813)

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 5](#_Toc73918814)

[1.1 Предметная область 5](#_Toc73918815)

[1.2 Обзор аналогичных проектов 5](#_Toc73918816)

[1.3 Выбор языка и среды программирования 6](#_Toc73918817)

[1.4 Постановка задачи 8](#_Toc73918818)

[2 ОПИСАНИЕ ПРОГРАММЫ 9](#_Toc73918819)

[2.1 Общие сведения 9](#_Toc73918820)

[2.2 Процесс разработки пользовательского интерфейса 9](#_Toc73918821)

[2.3 Игровой процесс 10](#_Toc73918822)

[2.3.1 Структура классов 11](#_Toc73918823)

[2.3.2 Логика взаимодействия объектов 12](#_Toc73918824)

[2.3.3 Отслеживание движения объектов и действий игрока 15](#_Toc73918825)

[2.3.4 Окончание игры 18](#_Toc73918826)

[2.3.5 Музыкальное сопровождение 19](#_Toc73918827)

[ЗАКЛЮЧЕНИЕ 20](#_Toc73918828)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 21](#_Toc73918829)

# ВВЕДЕНИЕ

Современный человек ежедневно взаимодействует с компьютерными технологиями. С развитием технологий увеличивается и уровень взаимодействия. Одним из самых распространенных проявлений взаимодействия являются компьютерные игры [5]. В настоящее время игровая индустрия активно развивается. Каждый год на рынок выпускаются десятки тысяч новых игр, которые расходятся миллионами копий по всему миру и приносят десятки миллиардов долларов игровой индустрии [3]. Компьютерные игры стали неотъемлемой частью жизни людей разных лет. В отличие от других видов развлечений, компьютерные игры более доступны для конечного пользователя.

Одним из наиболее популярных жанров компьютерных игр являются аркады, подразумевающие короткий по времени интенсивный игровой процесс. Программы отличает всегда понятный сюжет, возможность играть бесконечно долго, получать бонусы и награды. Примером этого жанра является игра на телефон под названием «Doodle Jump», выпущенная компанией Lima Sky в 2010 году [6]. Своей простотой она завоевала свою популярность.

Целью выполнения курсовой работы является разработка компьютерной игры «Doodle Jump». Актуальность данной разработки заключается в создании нового улучшенного варианта игры, в котором учитываются недостатки аналогов, а также адаптации игры для компьютера. Игра имеет яркий интерфейс и понятный геймплей, поэтому интересна для игроков разных возрастных категорий.

Объектом исследования являются игры, разработанные на языке C++.

Предметом исследования являются аркадные игры, цель которых подняться как можно выше, прыгая с платформы на платформу.

# АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## Предметная область

«Doodle Jump» простая и веселая игра, рассчитанная на одного игрока. Главная цель игры — помочь «Дудлеру» (Doodle) добраться как можно выше по платформам и набрать как можно большее количество очков. Героем можно управлять с помощью кнопок на клавиатуре (влево, вправо, пробел). В игре представлен простой, но привлекательный интерфейс.

Согласно правилам, игра заканчивается, когда игрок падает (по достижении нижней части экрана), попадает на монстра или на него падает платформа [6].

Во время игры счёт отображается в верхнем левом углу экрана, а затем наивысший результат сохраняется.

Игра имеет большую популярность среди пользователей, поэтому ее реализация уместна.

## Обзор аналогичных проектов

Игр типа Doodle Jump множество. Задача в таких играх, как правило, сводится к поиску пути наверх. Чем выше находится игрок, тем сложнее сохранить равновесие и продолжить игру.

Одним из представителей таких игр является игра «Jelly Jump» [7]. Данная игра реализована 2D графике. Положительными чертами игры является то, что у нее есть платформы, которые открываются и могут столкнуть игрока вниз, что делает игру более интересной. Для того чтобы разнообразить игровой процесс, предлагается собирать в процессе прохождения капли, которые станут материалом для создания нового игрового куба.

Еще одним аналогом является игра «Beat Stomper» [8]. Это музыкальная аркада, где вы управляете квадратом и стараетесь подняться как можно выше. Положительными чертами игры являются интересное музыкальное сопровождение, а также возможность переключаться между уровнями.

Еще одним представителем является игра «Happy Jump» [9]. Основными преимуществами игры является наличие сюжета и миссий в игре, магазина, в котором за игровые монеты можно изменить внешний вид главного героя. Однако игра имеет ряд недостатков, среди которых реклама, которая появляется посреди игры, отсутствие рекордов и адаптации для пользователей компьютера.

Еще одним примером аркадных приложений является игра «Go Up» [10]. Смысл игры прост: необходимо провести мячик, который находится у подножия ступенек. Движется мяч самостоятельно, от игрока лишь требуется вовремя нажимать по экран, заставляя мяч подпрыгивать и забираться на ступеньку выше. Если этого не сделать, мячик врежется в стену, а игру придется начинать заново. Преимуществами данной игры являются красочный интерфейс, простое управление, разнообразие игровых мячей. К недостаткам можно отнести наличие рекламы и отсутствие реализации для пользователей компьютеров.

В процессе разработки игры «Doodle Jump» учтены все преимущества и недостатки аналогов, добавлены новые возможности для игрока, чтобы сделать игру уникальной.

## Выбор языка и среды программирования

Для написания игры «Doodle Jump» был выбран язык C++, поскольку имеются практические навыки в разработке приложений на этом языке программирования. Язык программирования С++ представляет высокоуровневый компилируемый язык программирования общего назначения со статической типизацией, который подходит для создания самых различных приложений [1]. На сегодняшний день С++ является одним из самых популярных и распространенных языков. Преимуществом этого языка для создания игры является то, что она объединяет преимущества других распространенных языков программирования и имеет ряд собственных особенностей:

* поддержка различных парадигм программирования;
* поддержка объектно-ориентированного программирования;
* высокая скорость исполнения программы.

В качестве среды программирования было выбрано Qt. Это кроссплатформенный инструментарий разработчика прикладного программного обеспечения, широко используемый для создания графических интерфейсов.

Первым преимуществом данной среды является возможность писать код на языке C++, поскольку именно этот язык программирования был выбран для написания игры. Среди других основных преимуществ Qt можно отметить следующие:

* понятный и доступный интерфейс;
* возможность создания 2D игр;
* возможность тестирования игры при ее создании;
* наличие понятной документации;
* наличие удобного взаимодействия (сигналы/слоты).

Qt позволяет использовать пользовательские изображения, что значительно упрощает создание графики игры, и включает в себя множество классов для создания графического интерфейса, а также все основные классы, которые могут потребоваться при разработке прикладного программного обеспечения. Является полностью объектно-ориентированным, расширяемым и поддерживающим технику компонентного программирования [4].

Итак, C++ и инструментарий Qt ­­ — лучший выбор для написания игры «Doodle Jump», поскольку этот язык программирования имеет перечень преимуществ, которых нет в других высокоуровневых языках программирования.

## Постановка задачи

Задачей данного проекта является написание игры под названием «Doodle Jump». Игра будет создана в среде программирования Qt с использованием языка программирования C++. Управление будет осуществляться с помощью клавиатуры.

Главная цель игры — добраться как можно выше по платформам и набрать как можно большее количество очков. Игрок передвигается, прыгая с платформы на платформу и избегая различных препятствий. Для удобства прохождения игры поставлены следующие задачи:

* создать меню для начала игры;
* добавить музыкального сопровождения;
* добавить возможность поставить игру на паузу в любой момент;
* создать приятный графический интерфейс;
* добавить различные виды платформ;
* добавить различные виды препятствий и поощрений для игрока;
* сохранять лучший счет игрока;
* добавить возможность стрелять и уничтожать препятствия.

Очки будут начисляться с повышением высоты с учетом типа платформы и бонусов игрока. Игра должна быть простой и захватывающей.

# ОПИСАНИЕ ПРОГРАММЫ

## Общие сведения

Игра предназначена для развлечения и хорошего времяпрепровождения.

«Doodle Jump» – это яркая аркада на компьютер о приключениях прыгающего существа. Главное действующее лицо, Doodle, будет осуществлять свой длинный путь вверх, а игрок будет ему помогать. Он передвигаться, прыгая с платформы на платформу, но платформы бывают разные: двигающиеся, засасывающие, толкающие. Есть много препятствий: приведения, падающие платформы. Есть бонусы: пружины, объекты, повышающие счет.

## Процесс разработки пользовательского интерфейса

Для создания приятного интерфейса были выделены следующие критерии:

* интерфейс должен быть простым и понятным;
* интерфейс должен быть удобным в использовании;
* интерфейс должен быть приятным на вид, чтобы было удобно играть.

Главное окно любого Qt приложения – это класс-наследник QMainWindow [4]. Класс MainWindow представляет собой основное окно приложения.

При запуске игры пользователь видит главное игровое меню (рис. 1). Пользователь должен нажать на кнопку «play» для запуска новой игры. Кнопка «play» представляет собой объект класса Button, который содержит следующие функции для обработки событий кнопки:

* mousePressEvent() вызывается при клике на кнопку, срабатывает сигнал clicked, соединенный со слотом newGame();
* hoverEnterEvent() вызывается при наведении курсора мыши на кнопку, которая в этот момент меняет цвет на красный;
* hoverLeaveEvent() вызывается, когда курсор мыши покидает кнопку, она меняет свой цвет на первоначальный.



Рисунок 1 – Игровое меню

Игровое меню сделано максимально понятно и просто, чтобы пользователь не запутался в большом количестве кнопок в самом начале игры.

## Игровой процесс

При создании данной игры мы будем придерживаться ООП. Объектно-ориентированное программирование — методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования [2]. Каждый класс будет храниться в отдельном файле, что позволит легко удалять старые и добавлять новые функции в нашу программу.

После нажатия кнопки «play» запускается новая игра. Окно меню меняется на игровое окно при помощи слота newGame(). Он создает новый объект класса GameControl, который контролирует поведение всех игровых объектов. Каждый объект является представителем отдельного класса.

### Структура классов

Все классы, представляющие собой игровые объекты, являются наследниками класса QGraphicsPixmapItem. Это нужно для того, чтобы поместить в объект класса пользовательское изображение для создания графического интерфейса.

Игровые объекты содержаться в следующих классах:

* Platform — базовый класс для всех видов платформ в игре. Имеет классы-наследники SimplePlatform (статическая платформа зеленого цвета), DriveHorizontalPlatform (горизонтально движущаяся платформа), MovingPlatform (голубая платформа, которая сильнее отталкивает игрока), StopPlatform(оранжевая платформа, которая не дает игроку оттолкнуться).
* Threat — базовый класс для всех существующих видов угроз в игре. Имеет классы-наследники Ghost (приведение), FallingPlatform (платформа, движущаяся вертикально вниз). Встреча игрока с угрозой означает окончание игры. Игрок может уничтожить угрозу, нажав на клавишу пробел и выстрелив в нее.
* Attack — класс для генерации выстрела. Содержит единственный метод move(), позволяющий двигать пулю от выстрела по игровому экрану. При выходе за пределы окна объект уничтожается.
* GameHelper — базовый класс для всех бонусов игрока. Имеет классы-наследники SpringBoots (пружины, позволяющие игроку выше прыгать) и Multiplier (объекты этого класса увеличивают количество очков в определенное количество раз при прыжке с платформы на платформу).
* PlayDoodle — класс, представляющий объект игрока Doodle. В нем описаны его взаимодействия со всеми остальными классами, прыжки выстрелы и пр.

Такая структура программы помогает с легкостью обновлять и улучшать игру в дальнейшем. Для добавления новых объектов нужно создать класс самого объекта и прописать логику его взаимодействия в классах PlayDoodle и GameControl.

### Логика взаимодействия объектов

Взаимодействие игрока Doodle с объектами игрового поля находится в классе PlayDoodle. При создании объекта данного класса добавляется изображене игрока Doodle (рис. 2).



Рисунок 2 — Игрок Doodle

При помощи слотов moveLeft() и moveRight(), которые соединены с сигналом нажатия на клавиатуре стрелочки влево и вправо, изображение будет меняться: игрок будет смотреть влево или вправо соответственно. Это позволяет создать более реалистичный интерфейс игры.

Слоты moveUp() и moveDown() отвечают за физику игры. При движении влево и вправо Doodle прыгает по параболе. Переменная gravity отвечает за степень притяжения игрока к поверхности. Тогда код этих слотов выглядит следующим образом:

void PlayDoodle::moveUp()

{

deltaY-=gravity;

setPos(x(), y()-deltaY);

if(deltaY<=0){

setMoveVertDirection(DOWN);

deltaY=0;

}

}

void PlayDoodle::moveDown()

{

deltaY+=gravity;

setPos(x(), y()+deltaY);

if((y()+playerHeight)>VIEW\_HEIGHT){

emit controller.gameOver();

}

boostFactor=1;

}

Переменная deltaY — высота прыжка от поверхности платформы. При достижении игроком края окна происходит окончание игры и вызов соответствующего меню.

В классе PlayDoodle имеются 2 переменные, отвечающие за направление движения:

currentDirectionотвечает за вертикальное направление движения игрока, изменяется при помощи функции setMoveVertDirection() на заданное направление;

movingDirectionотвечает за горизонтальное направление движения, изменяется при помощи функции moveDirection() на заданное направление, используется для определения изображения игрока. В зависимости от значения (RIGHT или LEFT) игрок будет двигаться вправо или влево соответственно.

Кроме того, игрок взаимодействует с различными объектами игрового окна. За это отвечают функции meetPlatform(), meetThreats(), meetMultiplier(). Рассмотрим функцию взаимодействия игрока с платформами, взаимодействие с угрозами и бонусами происходит аналогично.

bool PlayDoodle::meetPlatform()

{

QList<QGraphicsItem \*> collisions=this->collidingItems();

if(scene()->collidingItems(this).isEmpty()){

return false;

}

else{

foreach(QGraphicsItem \*collidingItem, collisions){

if(collidingItem->data(TYPE)==PLATFORM &&

collidingItem->y() - playerHeight + DISPLACEMENT>=y()){

distanceToGround=VIEW\_HEIGHT-collidingItem->y();

previousY=static\_cast<Platform \*>(collidingItem)->Y();

if(collidingItem->data(PLATFORM\_TYPE)==STOPING)

boostFactor=0;

else if(collidingItem->data(PLATFORM\_TYPE)==MOVING)

boostFactor=2;

else if(collidingItem->data(PLATFORM\_TYPE)==DRIVE)

boostFactor=1;

else

boostFactor=1;

return true;

}

}

return false;

}

}

При перемещении игрока вдоль игрового окна, создается список объектов, изображения которых пересекаются. Если этот объект является платформой, осуществляется проверка допрыгнет ли игрок до этой платформы. Далее определяется тип платформы и устанавливается boostFactor, степень толчка от платформы.

Функции взаимодействия объектов возвращают значение true или false. Эти значения используются в слоте advance(), который позволяет обновлять состояние объектов во время игры. Если функция meetPlatform() возвращает true, вызывается функция jump(), отвечающая за прыжок игрока. Если функция meetThreats() возвращает true, текущая игра завершается и вызывается окно окончания игры.

### Отслеживание движения объектов и действий игрока

Контроль над всеми объектами, их создание, отслеживание действий игрока происходит в классе GameControl. Он содержит методы инициализации, генерации объектов, а также их движения:

* initPlayer() отвечает за создание игрока;
* initGhost() создает объект приведения;
* initPlatform() создает группу платформ разного типа;
* initFallingPlatform() создает объект падающей платформы;
* initMultiplier() создает объект бонусов игрока.

Слоты generatePlatform(), moveGhost(), movePlatform(), generateGhost(), moveFallingPlatform(), generateFallingPlatform(), generateMultiplier() связаны с объектом QTimer и вызываются каждые 16 миллисекунд для генерации новых объектов во время движения игрока по экрану.

Управление действиями Doodle происходит при помощи нажатия клавиш на клавиатуре. Обработка нажатия клавиш происходит в функциях handleKeyPressed() и handleKeyRelease(). Если пользователь нажимает клавишу влево или вправо, направление горизонтального движения объекта класса playDoodle становится левым или правым соответственно. Когда пользователь отпускает эти клавиши, Doodle прекращает движение в сторону и прыгает на одном месте. При нажатии клавиши Esc игра становится на паузу, все слоты и сигналы отсоединяются друг от друга, при повторном нажатии клавиши Esc происходит связывание всех слотов и сигналов, игра продолжается с того места, на котором остановилась. При нажатии клавиши пробел происходит выстрел:

case Qt::Key\_Space:

if(paused==false){

QPixmap pixmap;

bool success = pixmap.load(PLAYER\_SHOOT\_PATH);

if (!success){

exit(-1);

}

playDoodle->setPixmap(pixmap.scaledToWidth(PLAYER\_WIDTH));

attack=new Attack();

attack->setPos(playDoodle->x()-5,playDoodle->y()-

playDoodle->getPlayerHeight()/2);

scene.addItem(attack);

}

break;

Кроме того, в классе GameControl осуществляется подсчет общего количества очков в игре. Оно зависит от переменной deltaY, от типа платформы, от которой отталкивается игрок, и от бонусов игрока. Общий счет находится в левом верхнем углу экрана и переписывается каждый раз после прыжка игрока. Игровой интерфейс можно увидеть на рисунке 3.

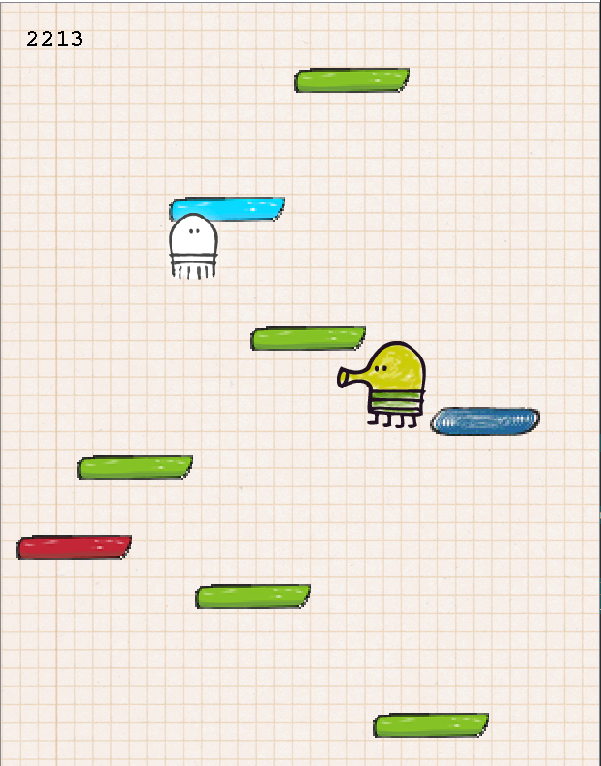


Рисунок 3 — Игровой процесс

Класс GameControl отвечает за движение угроз, платформ в кадре, а также за движение камеры вместе с Doodle. Во время прыжка все объекты должны двигаться вместе с игроком, чтобы создавался реалистичный эффект перемещения. Рассмотрим функцию движения камеры вместе с игроком:

void GameControl::moveCamera() {

if(playDoodle->y()<VIEW\_HEIGHT/2-playDoodle->getPlayerHeight()){

playDoodle->setY(VIEW\_HEIGHT/2-playDoodle->getPlayerHeight());

std::vector<Platform\*>::iterator it=platforms.begin();

for(int i=0;it!=platforms.end();it++, i++){

if(playDoodle->getCurrentDirection()==UP){

platforms[i]->setY(platforms[i]->y()+playDoodle->getDeltaY());

}

}

ghost->setY(ghost->y()+playDoodle->getDeltaY());

drivePlatform->setY(drivePlatform->y()+playDoodle->getDeltaY());

fallingPlatform->setY(fallingPlatform->y()+playDoodle->getDeltaY());

multiplier->setY(multiplier->y()+playDoodle->getDeltaY());

springBoots->setY(springBoots->y()+playDoodle->getDeltaY());

addScore();

}

}

Вместе с игроком на высоту его прыжка сдвигаются все объекты и счет игры. Таким образом, создается впечатление статичности заднего плана и движения игрока вверх.

Движение игровых объектов описано в отдельных слотах. Перемещение двигающейся платформы и приведения происходит одинаково. За это отвечают функции movePlatform() и moveGhost(). Как только объект достигает своего минимального или максимального положения по горизонтали, направление его движения меняется на противоположное. Если объект заходит за правую или левую границу окна, он перемещается на противоположную сторону окна. Перемещение падающей платформы происходи за счет увеличения координаты у. За это отвечает функция moveFallingPlatform().

### Окончание игры

Если игрок достигает нижнего края окна или попадает на угрозу, игра заканчивается и вызывается сигнал gameOver(), связанный со слотом gameOverSlot(). При вызове gameOverSlot() происходит отсоединение всех сигналов и слотов игры.

Игровое окно изменяется на окно окончания игры (рисунок 4).



Рисунок 4 ­­— Меню окончания игры

В этом окне в первой строке отображается счет игрока за текущую игру, во второй строке отображается лучший счет игрока. Лучший счет сохраняется с момента запуска игрового окна до момента его закрытия. Кроме того, в этом окне расположена кнопка «menu», которая переносит игрока в главное меню приложения для старта новой игры. Она является объектом класса Button, описанного ранее.

### Музыкальное сопровождение

Музыкальное сопровождение — одна из ключевых составляющих эмоциональной основы игр. Она помогает поддерживать интерес игроков, мотивировать на прохождение новых вершин и вовлекать в игру. Так, с помощью музыки мы можем самое главное — вызывать эмоции.

Поэтому для создания более захватывающего игрового процесса в приложение была добавлена музыка. С технической точки зрения звук подключен к проекту при помощи классов QMediaPlayer и QMediaPlaylist [4]. Для того, чтобы придать различную эмоциональную окраску разным этапам игры, мной было добавлено несколько звуковых эффектов.

Первая музыкальная композиция воспроизводится при запуске игры и ассоциируется с приливом сил, энергии и новыми свершениями. Вторая композиция воспроизводится при проигрыше во время вызова окна окончания игры и дает четкое представление о неуспешном прохождении игры.

При запуске основного меню приложения установлен циклический режим проигрывания аудиодорожки, а при открытии меню окончания игры — однократный.

# ЗАКЛЮЧЕНИЕ

В рамках данной работы была спроектирована и реализована компьютерная игра «Doodle Jump» в жанре аркада. Проект был написан на языке C++ в среде программирования Qt. Для достижения этой цели были выполнены следующие задачи:

* проведен анализ предметной области;
* проведен аналогичных проектов;
* проведен анализ и выбор средств реализации игры;
* описана концепция игры;
* разработан пользовательский интерфейс;
* реализована игра.

В ходе реализации потребовалось выполнить ряд задач, связанных с

созданием концепции и графической составляющей игры. К таким задачам

относится создание главного меню игры, главного персонажа, платформ, препятствий, меню окончания игры.

Игра был протестирован многими пользователями. Ошибок при этом выявлено не было. В перспективе планируется увеличить количество пользователей данной игры. Также возможно появление новых функций, что будет несложно реализовать, так как было использовано ООП.

Дальнейшее развитие данного игрового приложения может идти по нескольким направлениям:

* добавление различных звуков (прыжок, стрельба и т.п.);
* добавление новых видов платформ, бонусов и угроз;
* добавление новых миров в игре;
* добавление возможности играть по сети;

Достоинством этой программы является простота управления основным объектом, минимальные системные требования, достаточно интересная реализация графики с использованием графических примитивов.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Страуструп, Б. Язык программирования C++. Специальное издание /

Б. Страуструп. — Бином, 2008.

1. Лафоре, Р. Объектно-ориентированное программирование в С++. 4-е издание / Р. Лафоре. — СПб.: Питер, 2004.
2. Статья об истории развитии компьютерных игр. [Электронный ресурс] URL: <http://cpu3d.com/histgame/statya-ob-istorii-razvitiya-kompyuternyh>
3. Qt Documentation. [Электронный ресурс] URL: <https://doc.qt.io/>
4. Скалозуб И. Компьютерные игры как средство развития коммуникативных и личностных особенностей подростков // Молодой ученый, 2015
5. Doodle Jump Fandom. [Электронный ресурс] URL: <https://doodlejump.fandom.com/ru/wiki/Doodle_Jump>
6. Обзор Jelly Jump. [Электронный ресурс] URL: https://myoptimus.net/obzor-jelly-jump-igra-kotoraya-uvlechet-vas-na-do
7. Beat Stomper. [Электронный ресурс] URL: <http://odintap.ru/ios/item/4280-beat-stomper>
8. Игра Happy Jump. [Электронный ресурс] URL: <https://play.google.com/store/apps/details?id=com.noodlecake.happyjump&hl=ru&gl=US>
9. Обзор игры Go Up. [Электронный ресурс] URL: https://prodroiders.net/igry/arkady/go-up