Министерство образования и науки РФ

Севастопольский государственный университет

Кафедра ИС

КУРСОВАЯ РАБОТА ПО ДИСЦИПЛИНЕ

«ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ»

ПРОГРАММА «Platformer»

Пояснительная записка

Листов

ПРОВЕРИЛ

Ассистент кафедры ИС

( ) Н.Р. Головизнина

« » \_\_\_\_\_\_\_\_\_2015

РАЗРАБОТАЛ

Ст. гр. И-21д

( ) И.В. Поляков

« » \_\_\_\_\_\_\_\_\_2015

Ст. гр. И-21д

( ) Н.Е. Жихарев

« » \_\_\_\_\_\_\_\_\_2015

Севастополь

2015

# АННОТАЦИЯ

Данная пояснительная записка в полной мере описывает разработанный игровой продукт жанра Platformer, а также включает в себя проектное решение и программную реализацию проекта на платформе Unity. Помимо этого, документ описывает интерфейс пользователя и критерии качества программного продукта.

*Изм.*

*Лист*

*№ Документа*

*Подпись*

*Дата*

*Лист*

3

КУРСОВАЯ РАБОТА

*Разработал*

*Жихарев Н.Е. Поляков И.В*

*Руковод.*

*Забаштанский А.К.*

*Т.контр.*

*Н.контр.*

*Утв*

Platformer

*Лит.*

*Листов*

90

гр. И-21д

СОДЕРЖАНИЕ

[АННОТАЦИЯ 2](#_Toc419935781)

[ВВЕДЕНИЕ 4](#_Toc419935782)

[1. ПОСТАНОВКА ЗАДАЧИ 5](#_Toc419935783)

[1.1. Цель разработки 5](#_Toc419935784)

[1.2. Описательная постановка задачи 5](#_Toc419935785)

[2. ПРОЕКТНОЕ РЕШЕНИЕ 6](#_Toc419935786)

[2.1. Разработка и выделение объектов 6](#_Toc419935787)

[2.2. Диаграмма классов 8](#_Toc419935794)

[3. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ 9](#_Toc419935795)

[3.1. Обоснование выбора языка программирования 9](#_Toc419935796)

[3.2. Описание реализации основных классов и их методов 9](#_Toc419935797)

[3.2.1. Класс «Player» 9](#_Toc419935798)

[3.2.2. Класс «Item» 10](#_Toc419935799)

[3.2.3. Класс «Inventory» 11](#_Toc419935800)

[3.2.4. Класс «Camera2DFollow» 12](#_Toc419935801)

[3.2.5. Класс «Weapon» 13](#_Toc419935802)

[3.2.6. Класс «GameMaster» 15](#_Toc419935803)

[3.2.7. Класс «MoveAlien» 15](#_Toc419935804)

[3.2.8. Класс «PlatformerCharacter2D» 16](#_Toc419935805)

[3.3. Описание интерфейса пользователя 18](#_Toc419935806)

[3.4. Критерии качества программной системы 19](#_Toc419935807)

[3.5. Условия выполнения программы 21](#_Toc419935808)

[ЗАКЛЮЧЕНИЕ 22](#_Toc419935809)

[СПИСОК ЛИТЕРАТУРЫ 23](#_Toc419935810)

[ПРИЛОЖЕНИЕ А 24](#_Toc419935811)

[ПРИЛОЖЕНИЕ Б 44](#_Toc419935812)

# ВВЕДЕНИЕ

В рамках поставленной задачи по предмету «Объектно-ориентированное программирование» ведется разработка программного продукта под названием «Platformer» в соответствии с утвержденным преподавателем техническим заданием.

Разрабатываемый программный продукт можно отнести к категории игровых программных продуктов. В качестве основной парадигмы разработки был применен объектно-ориентированный подход (далее ООП). Применение ООП при проектировании программных продуктов со сложной архитектурой позволяет создать легкий для расширения и сопровождения проект высокого качества. Актуальность применения ООП обусловлена тем, что данный подход является одним из передовых в мировой IT индустрии, это обеспечивает широкий набор инструментов для разработки и снижает сложность изучения парадигмы.

Актуальность разработки игровых программных продуктов обеспечена большим спросом как на рынке персональных компьютеров, так и на рынке смартфонов. А наличие централизованного рынка реализации такого рода продуктов (Google Market) позволяет упростить процесс монетизации приложений.

Целью данного курсового проектирование является закрепление навыков разработки приложений с применением ООП, а также получение опыта проектирования сложных программ, содержащих множество классов и связей. Кроме того, важной составляющей является получение опыта объектного анализа и разработки с применением фреймворков.

В разделе «Проектное решение» описывается ход проектирования объектной иерархии, информационной модели и жизненного цикла программы, и объектов. В разделе «Программная реализация» содержится описание особенностей программной реализации разрабатываемого продукта, описание интерфейса и критерии качества данного программного продукта.

# ПОСТАНОВКА ЗАДАЧИ

## Цель разработки

Разработать игровую систему на базе ООП, тем самым закрепить навыки разработки приложений и получить навыки проектирования с применением языка UML.

## Описательная постановка задачи

Система моделирует взаимодействие игрока с различными объектами игрового мира.

Силы врага представляют собой юниты(существа обладающие примитивным искусственным интеллектом), целью которых является атака игрока. Юниты, как и игрок, имеют определенное количество очков здоровья.

Игрок управляет юнитом при помощи мыши и клавиатуры. Целью игрока является достижение конечной точки уровня путем преодоление препятствий. Препятствия представлены как в виде вражеских юнитов, так и в виде статических объектов, обладающих физическими свойствами (стены, ямы и т.д.). При этом игрок может уничтожать вражеских юнитов используя оружие. Оружие, доступное игроку, также обладает некоторыми свойствами: скорострельность и урон. Так, например, винтовка наносит больше урона, чем пистолет, однако скорость стрельбы у неё меньше.

Условие победы – достичь конечной точки уровня.

Условие поражения – очки здоровья игрока равны 0.

Входными данными является ввод с клавиатуры и мыши – размещение башен на карте, нажатие кнопок. Выходные данные выводятся на экран в игровое окно.

# ПРОЕКТНОЕ РЕШЕНИЕ

## Разработка и выделение объектов

Для реализации задачи, поставленной в курсовом проектировании, необходимо выполнить декомпозицию системы на объекты.

### Класс «Player»

Данный класс содержит в себе описание всех доступных для игрока действий, реализует анимацию игрового персонажа и контролирует аудио эффекты персонажа. За характеристики игрока отвечает вложенный класс PlayerStats, он описывает количество очков жизни, которыми будет обладать игрок с начала игры, а также их максимальное количество.

Также данный класс содержит методы для обработки взаимодействия игрока с различными игровыми объектами и определяет условие смерти и появления персонажа.

* + 1. Класс «Item»

Данный класс описывает абстракцию «Предмет». В игре есть несколько классов предметов: оружие и не оружие. При этом каждый из них обладает своим уникальным номером, типом, именем, и свойством, показывающим является ли предмет активным в данный момент.

* + 1. Класс «Inventory»

Данный класс реализует абстракцию «Инвентарь». Инвентарь необходим пользователю для содержания нескольких видов различных вещей. При этом класс предоставляет методы для удаления и добавление вещей в инвентарь, для переключения между вещами и для получения ID активного предмета.

### Класс «Camera2DFollow»

Данный класс описывает игровую камеру, которая следует за игроком по мере его продвижения по уровню. Камера обладает свойством цели – объект за которым она движется, а также несколькими свойствами, определяющими её положение в следующий момент времени – насколько и в какую сторону необходимо сместиться.

### Класс «Weapon»

Данный класс описывает абстракцию «Оружие». Он содержит такие свойства как: скорость стрельбы, количество патронов, тип оружия, размер обоймы, а также содержит модель и звуки конкретного типа оружия. Содержит методы, реализующие стрельбу из данного типа оружия, его перезарядку и действия при отсутствии патронов.

### Класс «GameMaster»

Данный класс отвечает за загрузку и удаление объектов игрового мира, а также управляет воскрешением персонажа и его начальными предметами.

### Класс «MoveAlien»

Данный класс реализует поведение одного из вражеских юнитов. Содержит методы, отвечающие за появление и перемещение юнита по игровому миру, а также свойства о здоровье и скорости.

### Класс «PlatformerCharacter2D»

Данный класс представляет собой основной контроллер игрового мира. Он устанавливает такие параметры, как порядок переходов и анимации игрового персонажа, настраивает маркеры для проверки коллизии, устанавливает клавиши управления персонажем, а также с помощью специальных методов определяет его поведение для различных состояний(бег, ходьба, выбор оружия).

## Диаграмма классов

# ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

## Обоснование выбора языка программирования

Выбирая в качестве основного языка разработки язык C# и используя игровой движок Unity с интегрированной средой разработки MonoDevelop, мы руководствовались в первую очередь тем, что C# является наиболее актуальным и развивающимся императивным языком программирования, реализующим все концепции ООП в полной мере и позволяющий легко представить спроектированную архитектуру в виде программного кода. В тоже время движок Unity позволяет написать качественное игровое приложение, которое будет правильно функционировать на всех актуальных операционных системах, в том числе на Android.

Кроме того, для платформы Unity разработано множество библиотек и модулей, позволяющих снизить объем программного и ускорить процесс разработки.

Хорошая интеграция среды разработки MonoDevelop с игровым движком значительно упрощает навигацию по проекту и разработку новых программных модулей с последующей интеграцией в систему. Таким образом, для выполнения поставленной задачи, целесообразным является использование выбранных языка программирования, игрового движка и среды разработки.

## Описание реализации основных классов и их методов

### Класс «Player»

Данный класс описывает атрибуты и свойства присущие игровому персонажу, хранит изображение и звуки персонажа.

Максимальное и текущее количество очков жизни, которые могут быть у игрока:

public int maxHealth = 100;

public int health = 100;

Позиция игрока по оси Y которая для него является смертельной (таким образом проверяется, не упал ли игрок за карту):

public int fallBoundary = -20;

Задержка до воскрешения:

private float delayBeforeDead = 1f;

Задержка при попадании:

private float takingDamageDelay = 0.3f;

Метод обновления состояния персонажа за игровую единицу времени:

void Update ();

Метод, описывающий получение урона персонажем. Выполняет проверку на текущее количество хп и вычитает полученный урон, если после этого очки жизни персонажа стали меньше или равны 0, то вызывается метод смерти игрока.

public void DamagePlayer (int damage);

Метод, проверяющий столкновение персонажа с вражеским героем. Если столкновение произошло, то вычитается 10 очков жизни. В качестве параметра передается коллизия вражеского юнита.

void OnCollisionEnter2D(Collision2D col);

Метод, который вызывается если у игрока закончились очки жизни, при этом состояние игрового персонажа переходит в анимацию смерти, а менеджер игрового мира отправляет персонажа в начальную точку уровня.

public IEnumerator KillPlayer();

Метод, реализующий механику получения урона персонажем. Он вызывает соответствующие аудио эффекты и производит задержку.

public IEnumerator TakingDamage();

### Класс «Item»

Класс реализует абстрактную сущность «Предмет».

Перечисление типов предметов:

public enum ItemType

{

Weapon,

NotWeapon

}

Уникальный номер предмета:

public int itemID;

Название предмета:

public String itemName;

Значение предмета:

public int itemValue;

Тип предмета:

public ItemType itemType;

Состояние активности предмета:

public bool isActive;

Конструктор выполняющий инициализацию всех полей:

public Item (int \_itemID, ItemType \_itemType, String \_itemName, int \_itemValue, bool \_isActive);

### Класс «Inventory»

Данный класс реализует сущность «Инвентарь» и действия с ним. Класс хранит список сущностей «Предмет» и реализует набор действий над ними.

Список предметов:

public List<Item> itemList;

Загрузка предметов. Данный метод инициализирует список предметов в инвентаре, а также загружает туда несколько типов оружия:

void Awake();

Выставить предмет с заданным в параметре именем активным:

public void SetActiveByName(System.String itemName)

Выставить предмет с заданным в параметре номером активным:

public void SetActiveByID(int itemID)

Добавить предмет:

public void AddItem(Item newItem)

Выставить значение предмета по имени заданному параметром. При этом новое значение также передается параметром:

public bool SetValueByName(System.String itemName, int itemValue)

Получить значение активного предмета:

public int GetActiveItemValue ()

Получить название активного предмета:

public System.String GetActiveItemName()

Удалить предмет по имени заданному в качестве параметра:

public bool DeleteItemByName(System.String itemName)

Проверить, имеется ли предмет с именем заданным в качестве параметра в инвентаре:

public bool IsExist (System.String itemName)

### Класс «Camera2DFollow»

Данный класс описывает игровую камеру, которая следует за игроком по мере его продвижения по уровню. При этом камера хранит координаты цели, за которой ей необходимо следовать, а также значения смещения и задержки.

Координаты цели, за которой необходимо следовать:

public Transform target;

Значения затухания движения при приближении к цели:

public float damping = 1;

Фактор смещения вперед:

public float lookAheadFactor = 3;

Скорость возврата при смещении:

public float lookAheadReturnSpeed = 0.5f;

Граница для смещения:

public float lookAheadMoveThreshold = 0.1f;

Ограничение по позиции Y:

public float yPosRestriction = -1;

Текущее ускорение:

private Vector3 currentVelocity;

Последняя позиция цели:

private Vector3 lastTargetPosition;

Частота обновления позиции камеры:

float nexTimeToSearch = 0;

Метод, выполняющий инициализацию основных полей класса:

private void Start();

Метод, выполняющий обновление положения камеры. При этом если игрок не был найден, то камера возвращается в начало, иначе выполняется переход к новой позиции игрока.

private void Update();

Метод, запрашивающий менеджер объектов игрового мира о наличии игрока:

void FindPlayer();

### Класс «Weapon»

Класс реализует абстракцию «Оружие» и содержит основные параметры, такие как скорострельность, количество патронов

Тип оружия:

public System.String weaponType;

Количество патронов:

private int bulletsCount;

Скорострельность:

public float fireRate = 0;

Урон от оружия:

public float damage = 10;

Количество патронов:

public int bulletsInMagazine = 0;

Предметы, реагирующие на выстрелы:

public LayerMask whatToHit;

Также класс хранит графические ресурсы, как например звуки, изображение текущего оружия, звуки и родительский инвентарь.

public AudioClip shootingSound;

public AudioClip outOfAmmoSound;

public AudioClip reloadingSound;

public Transform BulletTrailPrefab;

public Transform MuzzleFlashPrefab;

Метод, выполняющий инициализацию основных полей объекта, в частности выставляет определенное количество патронов в зависимости от типа оружия:

void Awake()

Метод, выполняющий обработку стрельбы, и осуществляющий ряд проверок: была ли нажата правая кнопка для проверки и находится ли игрок в нужном положении. Также этот метод осуществляет связь между нажатием определенных клавиш и действиями (стрельба, перезарядка).

void HandleShooting()

Метод реализующий стрельбу. Выполняет проверку, есть ли в магазине оружия боеприпасы, а также реализует процесс создания пуль и движения их в направлении прицела.

void Shoot ()

Метод осуществляющий обработку события отсутствия патронов в магазине:

void NoAmmmoLeft()

Метод, реализующий обработку перезарядки магазина в оружии:

void HandleReloading ()

Метод, реализующий процесс перезарядки магазина:

IEnumerator Reloading()

### Класс «GameMaster»

Данный класс реализует управление различными объектами в игровом в мире, в том числе игрока.

Позиция игрока:

public Transform playerPrefab;

Точка воскрешения игрока:

public Transform spawnPoint;

Задержка перед воскрешением:

public int spawnDelay = 2;

Инвентарь игрока:

private Inventory inventory;

Метод, отвечающий за инициализацию класса:

void Start ()

Метод, отвечающий за возрождение игрока, в случае если он погиб:

public IEnumerator RespawnPlayer ()

Метод, уничтожающий объект игрока:

public static void KillPlayer(Player player)

Метод, выполняющий обновление графического интерфейса игры:

void OnGUI()

### Класс «MoveAlien»

Данный класс реализует поведение одного из вражеских юнитов, а точнее зеленого движущегося инопланетянина.

Скорость движения юнита:

public float moveSpeed = 8.0f;

Направление движения:

private float direction = -1.0f;

Количество очков здоровья:

private float health;

Анимация юнита:

private Animator anim;

Метод, выполняющий инициализацию анимации юнита, а также выставляет начальную скорость бега в значение 10:

void Awake ()

Метод выполняет обновление юнита, вызывая метод бега:

void Update ()

Метод бега, выполняет перемещение юнита на расстояние равное его скорости умноженной на скорость:

void Run ()

Метод остановки игрового юнита. Скорость перемещения юнита становится равной нулю:

void Stop()

Метод, выполняющий проверку «жив ли юнит?» путем сравнения его количества жизней с нулем. Если юнит оказался мертв, то он удаляется из памяти:

void isDead()

### Класс «PlatformerCharacter2D»

Данный класс представляет собой основной контроллер игрового мира. Он устанавливает множество параметров игрового мира, настраивает анимацию игрока, а также определяет события и состояния в которых может находиться игровой юнит.

Максимальная скорость, с которой игрок может перемещаться вдоль оси оХ:

private float maxSpeed = 10f;

Сила, с которой игрок прыгает:

private float jumpForce = 400f;

Параметр, определяющий можно ли управлять героем во вермя прыжка:

airControl = false;

Объекты, которые являются землей:

private LayerMask whatIsGround;

Параметр, определяющий стоит игрок на земле или нет:

public bool grounded = false;

Параметр, определяющий в какую сторону смотрит игрок:

public bool facingRight = true;

Параметр, определяющий целится ли игрок в данный момент:

public bool isAiming;

Анимация игрока:

public Animator anim;

Коллизия игрока:

private BoxCollider2D boxCol;

private CircleCollider2D circleCol;

Метод, выполняющий инициализацию основных полей, в том числе анимации, инвентаря, различных видов оружия и коллизии игрока:

private void SetUpReferences()

Метод, выполняющий обновление анимации игрока:

private void FixedUpdate()

Метод обновления событий:

public void Update()

Метод, реализующий перемещение игрока в пространстве. Параметры указывают на возможные положения, в которых игрок может передвигаться:

public void Move(float move, bool crouch, bool jump)

Метод, меняющий положение руки с оружием в зависимости от положения персонажа:

private void Flip()

Метод, реализующий прицеливание по нажатию правой кнопки мыши:

private void Aiming()

Метод, разрешающий произвести стрельбу из оружия, если все параметры, необходимые для этого верны (у персонажа есть оружие и он находится в положении из которого можно стрелять):

public void ShootingEnabled()

Метод, запрещающий вести стрельбу:

public void ShootingDisabled()

Метод, отображающий в руке персонажа активное оружие:

public void ShowActiveWeapon()

Метод, реализующий выбор оружия по нажатию клавиш от 1 до 3, если игрок не прицеливается:

private void WeaponChoose()

Метод, запрещающий использовать все виды оружия:

private void DisableAllWeapons()

Методы, ставящие активным оружием пистолет, винтовку и автомат:

private void SetPistol()

private void SetRifle()

private void SetMachineGun()

Метод, скрывающий отключенные предметы:

private void HideDisabledItems ()

## Описание интерфейса пользователя

Игровой интерфейс представлен достаточно просто чтобы позволить пользователю без особых усилий разобраться с управлением и механикой игры. На рисунке 1 представлено основное окно игрового процесса.



Рисунок 3.1 – Внешний вид окна игрового процесса

На рисунке выше цифрами подписаны следующие компоненты игрового мира:

1. Интерфейс оружия. Как видно на нем располагается изображение текущего оружия и количество патронов – сколько их в обойме (цифра слева) и сколько их всего у игрока;
2. Статическое игровое препятствие, для преодоление которого игроку необходимо подпрыгнуть;
3. Статический вражеский юнит. Не перемещается, но если игрок попадает в его поле зрения, то начинает вести огонь. Также этот юнит подлежит взлому по нажатию клавиши «Е»;
4. Движущийся вражеский юнит. Если игрок попадает в поле зрения врага, то юнит начинает стремительное движение в сторону игрока и при столкновении наносит 10 единиц урона;
5. Игровой персонаж. Обладает способностями ходить, бегать, стрелять и прыгать. Также обладает полоской жизни, находящейся над головой персонажа;
6. Предметы, которые можно поднять. Существуют два различных типа: аптечки и патроны. Данные объекты поднять можно лишь в случае, если запах здоровья (или патронов) не максиальный.

## Критерии качества программной системы

Качество программного продукта было оценено по следующим критериям:

1. Соответствие объектной модели. Приложение разработано согласно всем требованиям и концепциям объектно-ориентированного подхода. При разработке были применены все основополагающие принципы ООП: полиморфизм, наследование и инкапсуляция.
2. Структурированность. Исходный код приложения имеет хорошую структурированность благодаря использованию объектного подхода. Сущности программы отделены друг от друга и ограничены различными уровнями абстракции. Все подсистемы, модули и отдельные сущности представлены в виде классов.
3. Расширяемость. Данное приложение легко дополнить и снабдить дополнительным функционалом благодаря качественной структуре проекта и удобным механизмам интеграции различных модулей в единую систему.
4. Поддерживаемость. Код программного продукта легко сопровождать и поддерживать, так как он оформлен по единому стилю согласно руководству по оформлению программного кода от Google.
5. Эффективность использование ресурсов. Разработанный менеджер игровых объектов позволяет отслеживать объекты, которые находятся за пределами игровой карты и удалять их из памяти, тем самым не засоряя память ненужными объектами.
6. Переносимость. Данное приложение возможно легко портировать на другие устройства благодаря применению игрового движка Unity, обеспечивающего кроссплатформенность приложений.
7. Удобность использования. Так как итоговый программный продукт обладает простым и понятным интерфейсом, не требующим много времени на освоение, то можно считать, что программа полностью отвечает данному критерию.
8. Надёжность. В ходе разработки и тестирования, не было замечено сбоев в работе движка, а все недоработки программной реализации были исправлены в ходе тестирования. Следовательно, разработанная программа достаточно надёжна.

## Условия выполнения программы

Для выполнения программы ЭВМ должна удовлетворять следующим системным требованиям:

Тактовая частота процессора не менее 700 ГГц

Видеокарта с объемом видеопамяти не менее 32 МБ

Не менее 512 МБ оперативной памяти

Операционная система: Windows 8/Windows 7/Windows XP/Linux/Mac OS

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проектирования были получены навыки проектирования сложных приложений, состоящих из множества компонентов и подсистема, а также были закреплены навыки написания приложений с использованием объектно-ориентированного подхода. Кроме того, были получены навыки работы с игровым движком Unity и изучены особенности разработки интерфейсов и игровой механики с помощью него.

При разработке приложения также были получены навыки тестирования и отладки приложения с применением систем автоматизированного тестирования и проверки отдельных модулей.

Была выполнена оценка качества программы по ряду критериев. Было определено, что разработанное программное решение является достаточно эффективным, надежным и способно удовлетворить требования среднестатистического пользователя, интересующегося игровыми приложениями.

Исходя из полученных результатов, можно сделать вывод что применением ООП имеет множество положительных сторон, в особенности при командной работе над сложным проектом. Применение языка UML для предварительного проектирования позволило до начала кодирования определить основные сущности системы, особенности их взаимодействия и способы коммуникации. В тоже время язык C# позволил в точности повторить архитектуру спроектированной системы и перенести её на движок Unity.

Таким образом можно считать, что использование объектно-ориентированного подхода при разработке данного приложения было оправдано, так как это позволило создать качественное приложение в установленные временные рамки. Благодаря этому можно считать, что цель курсового проекта была достигнута. Разработанная игровая система может применяться в игровой индустрии при соответствующей доработке.

# СПИСОК ЛИТЕРАТУРЫ

1. *Бабий А.А.* О некоторых характеристиках личной производительности труда программистов. // УСИМ, 1985, № 6, с. 17–19.

2. *Жоголев Е.А.* Технология программирования / Е.А. Жоголев. — М.: Научный мир, 2004. — 216 с.

3. *Д. Кнут.* Искусство программирования / Кнут Дональд. — М: Вильямс. 3-е издание. 2000.

4. Модели и структуры данных. Учеб. пособие / В.Д. Далека, А.С. Деревянко, О.Г. Кравец, Л.Е. Тимановская. — Харьков: ХГПУ, 2000.

4. *Буч Г.* Объектно-ориентированное проектирование с примерами применения / Г. Буч; Пер. с англ. — М.: Конкорд, 1992, — 519 с.

# ПРИЛОЖЕНИЕ А

Листинг программы

/\*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++

import javax.swing.\*;

import java.awt.\*;

public class Frame extends JFrame {

public static String title="Tower Defence";

public static Dimension size=new Dimension(700,550);

public Frame()

{

setTitle(title);

setSize(size);

setResizable(false);

setLocationRelativeTo(null);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

init();

}

public void init ()

{

setLayout(new GridLayout(1,1,0,0));

Screen screen=new Screen(this);

add(screen);

setVisible(true);

}

public static void main(String args[])

{ Frame frame=new Frame();

}

}

import java.awt.\*;

import java.awt.image.\*;

import java.io.File;

import javax.swing.\*;

public class Screen extends JPanel implements Runnable {

public Thread thread=new Thread(this);

public static int myWidth,myHeight;

public static int coinage=10, health=100;

public static boolean isFirst=true;

public static Room room;

public static int fpsFrame=0, fps=1000000;

public static Image [] tileset\_ground=new Image[100];

public static Image [] tileset\_air=new Image[100];

public static Image[] tileset\_res=new Image[100];

public static Image[] tileset\_mob=new Image[100];

public static boolean isDebug=false;

public static int killed=0,killsToWin=0,level=1,maxLevel=3;

public static int winTime=4000,winFrame=0;

public static boolean isWin=false;

public static Save save;

public static Point mse=new Point(0,0);

public static Store store;

public static Enemy [] mobs=new Enemy[100];

public Screen(Frame frame){

frame.addMouseListener(new KeyHandel());

frame.addMouseMotionListener(new KeyHandel());

thread.start();

}

public static void hasWon(){

if(killed==killsToWin){

isWin=true;

killed=0;

coinage=0;

}

}

public void define(){

room =new Room();

save=new Save();

store=new Store();

coinage=10;

health=100;

for (int i=0;i<tileset\_ground.length;i++)

{

tileset\_ground[i]=new ImageIcon("res/tileset\_ground.png").getImage();

tileset\_ground[i]=createImage (new FilteredImageSource(tileset\_ground[i].getSource(),new CropImageFilter(0,26\*i,26,26) ) );

}

for (int i=0;i<tileset\_air.length;i++)

{

tileset\_air[i]=new ImageIcon("res/tileset\_air.png").getImage();

tileset\_air[i]=createImage (new FilteredImageSource(tileset\_air[i].getSource(),new CropImageFilter(0,26\*i,26,26) ) );

}

tileset\_res[0]=new ImageIcon("res/cell.png").getImage();

tileset\_res[1]=new ImageIcon("res/health.png").getImage();

tileset\_res[2]=new ImageIcon("res/coin.png").getImage();

tileset\_mob[0]=new ImageIcon("res/mob.png").getImage();

save.loadSave(new File("save/mission"+level+".uixava"));

for (int i=0;i<mobs.length;i++){

mobs[i]=new Enemy();

}

}

public void paintComponent(Graphics g)

{

if(isFirst){

myWidth=getWidth();

myHeight=getHeight();

define();

isFirst=false;

}

g.setColor(new Color(70,70,70));

g.fillRect(0, 0, getWidth(), getHeight());

g.setColor(new Color(0,0,0));

g.drawLine(room.block[0][0].x-1, 0, room.block[0][0].x-1, room.block[room.worldHeight-1][0].y+room.blockSize+1);

g.drawLine( room.block[0][room.worldWidth-1].x+room.blockSize, 0, room.block[0][room.worldWidth-1].x+room.blockSize, room.block[0][room.worldWidth-1].y+room.blockSize);

g.drawLine(room.block[0][0].x, room.block[room.worldHeight-1][0].y+room.blockSize, room.block[0][room.worldWidth-1].x+room.blockSize, room.block[room.worldHeight-1][0].y+room.blockSize);

room.draw(g);

for (int i=0;i<mobs.length;i++){

if(mobs[i].inGame){

mobs[i].draw(g);

}

}

store.draw(g);

if(health<1){

g.setColor(new Color(255,20,20));

g.fillRect(0,0,myWidth,myHeight);

g.setColor(new Color(255,255,255));

g.setFont(new Font("Courier New",Font.BOLD,14));

g.drawString("Game Over!!!",10, 20);

}

if(isWin){

g.setColor(new Color(255,255,255));

g.fillRect(0, 0, getWidth(), getHeight());

g.setColor(new Color(0,0,0));

g.setFont(new Font("Courier New",Font.BOLD,14));

if(level==maxLevel){

g.drawString("You won the game",10, 20);

}else{

g.drawString("You won! Congratulation! Please wait for next level...",10, 20);

}

}

}

public int spawnTime=2400,spawnFrame=0;

public void mobSpawner(){

if(spawnFrame>=spawnTime)

{

for(int i=0;i<mobs.length;i++)

{

if(!mobs[i].inGame){

mobs[i].spawnMob(Value.mobGreeny);

break;

}

}

spawnFrame=0;

}

else {spawnFrame++;};

}

public void run(){

while(true){

if(!isFirst&&health>0&&!isWin){

room.physic();

mobSpawner();

for(int i=0;i<mobs.length;i++){

if (mobs[i].inGame){

mobs[i].physic();

}

}

}else{

if(isWin){

if(winFrame>=winTime){

if(level==maxLevel){

System.exit(0);

}else{

level++;

define();

isWin=false;

}

winFrame=0;

}

else{

winFrame++;

}

}

}

repaint();

try{

Thread.sleep(1);

}

catch(Exception e){}

}

}

}

import java.awt.\*;

public class Enemy extends Rectangle {

public int xC,yC;

public int health;

public int healthSpace=3,healthHeight=6;

public int mobSize=52;

public int mobWalk=0;

public int upward=0,downward=1,right=2,left=3;

public int direction=right;

public int mobID=Value.mobAir;

public boolean inGame=false;

public boolean hasUpward=false;

public boolean hasDownward=false;

public boolean hasLeft=false;

public boolean hasRight=false;

public Enemy(){

}

public void spawnMob(int mobID){

for (int y=0;y<Screen.room.block.length;y++){

if(Screen.room.block[y][0].groundID==Value.groundRoad){

setBounds(Screen.room.block[y][0].x,Screen.room.block[y][0].y,mobSize,mobSize);

xC=0;

yC=y;

}

}

this.mobID=mobID;

this.health=mobSize;

inGame=true;

}

public void deleteMob(){

inGame=false;

direction=right;

mobWalk=0;

Screen.room.block[0][0].getMoney(mobID);

}

public void looseHealth(){

Screen.health--;

}

public int walkSpeed=40,walkFrame=0;

public void physic(){

if(walkFrame>=walkSpeed){

if(direction==right){

x++;

}

else if(direction==upward){

y--;

}

else if (direction==downward)

{ y++;}

else if(direction==left){x--;}

mobWalk++;

if (mobWalk==Screen.room.blockSize){

if(direction==right){

xC++;

hasRight=true;

}

else if(direction==upward){

yC--;

hasUpward=true;

}

else if (direction==downward)

{ yC++;

hasDownward=true;

}else if (direction==left){

xC--;

hasLeft=true;

}

if(!hasUpward){

try{

if(Screen.room.block[yC+1][xC].groundID==Value.groundRoad){

direction=downward;

}

} catch (Exception e){}

}

if (!hasDownward){

try{

if(Screen.room.block[yC-1][xC].groundID==Value.groundRoad){

direction=upward;}

}catch (Exception e){}

}

if (!hasLeft){

try{

if(Screen.room.block[yC][xC+1].groundID==Value.groundRoad)

{

direction=right;}

}catch (Exception e){}

}

if (!hasRight){

try{

if(Screen.room.block[yC][xC-1].groundID==Value.groundRoad)

{

direction=left;}

}catch (Exception e){}

}

if(Screen.room.block[yC][xC].airID==Value.airCave)

{

deleteMob();

looseHealth();

}

hasUpward=hasDownward=hasLeft=hasRight=false;

mobWalk=0;

}

walkFrame=0;}

else walkFrame++;

}

public void looseHealth(int amo){

health-=amo;

checkDeath();

}

public void checkDeath(){

if(health==0){

deleteMob();

}

}

public boolean isDeath(){

if(inGame){

return false;

}

else{

return true;

}

}

public void draw(Graphics g){

if(inGame){

g.drawImage(Screen.tileset\_mob[mobID],x,y,width,height,null);

g.setColor(new Color(180,50,50));

g.fillRect(x,y-(healthSpace+healthHeight),width,healthHeight);

g.setColor(new Color(50,180,50));

g.fillRect(x,y-(healthSpace+healthHeight),health,healthHeight);

g.setColor(new Color(0,0,0));

g.drawRect(x,y-(healthSpace+healthHeight),health-1,healthHeight-1);

}

}

}

import java.awt.\*;

public class Block extends Rectangle{

public int groundID;

public int towerSquareSize=130;

public int airID;

public Rectangle towerSquare;

public int shotMob=-1;

public boolean shoting=false;

public int loseTime=100,loseFrame=0;

public Block(int x,int y, int width, int height,int groundID,int airID){

setBounds(x,y,width,height);

towerSquare=new Rectangle(x-(towerSquareSize/2),y-(towerSquareSize/2),width+(towerSquareSize),height+(towerSquareSize));

this.groundID=groundID;

this.airID=airID;

}

public void draw(Graphics g)

{

g.drawImage(Screen.tileset\_ground[groundID],x,y,width,height,null);

if(airID!=Value.airAir)

{

g.drawImage(Screen.tileset\_air[airID],x,y,width,height,null);

}

}

public void physic(){

if(shotMob!=-1&&towerSquare.intersects(Screen.mobs[shotMob])){

shoting=true;

}else{

shoting=false;

}

if(!shoting){

if(airID==Value.airTowerLaser){

for(int i=0;i<Screen.mobs.length;i++){

if(Screen.mobs[i].inGame){

if(towerSquare.intersects(Screen.mobs[i])){

shoting=true;

shotMob=i;

}

}

}

}

}

if(shoting){

if (loseFrame>=loseTime){

Screen.mobs[shotMob].looseHealth(1);

loseFrame=0;

}else{

loseFrame++;

}

if(Screen.mobs[shotMob].isDeath()){

shoting=false;

shotMob--;

Screen.killed++;

Screen.hasWon();

}

}

}

public void getMoney(int mobID){

Screen.coinage+=Value.deathReward[mobID];

}

public void fight(Graphics g){

if(Screen.isDebug){

if(airID==Value.airTowerLaser){

g.drawRect(towerSquare.x, towerSquare.y, towerSquare.width, towerSquare.height);

}

}

if(shoting){

g.setColor(new Color(255,255,0));

g.drawLine(x+(width/2), y+(height/2), Screen.mobs[shotMob].x+(Screen.mobs[shotMob].width/2),Screen.mobs[shotMob].y+(Screen.mobs[shotMob].height/2) );

}

}

}

import java.awt.\*;

import java.awt.event.\*;

public class KeyHandel implements MouseMotionListener,MouseListener {

public void mouseClicked(MouseEvent e) {

}

public void mouseEntered(MouseEvent e) {

}

public void mouseExited(MouseEvent e) {

}

public void mousePressed(MouseEvent e) {

Screen.store.click(e.getButton());

}

public void mouseReleased(MouseEvent e) {

}

public void mouseDragged(MouseEvent e) {

Screen.mse=new Point((e.getX())+((Frame.size.width-Screen.myWidth)/2),(e.getY())+((Frame.size.height-(Screen.myHeight))-(Frame.size.width-Screen.myWidth)/2));

}

public void mouseMoved(MouseEvent e) {

Screen.mse=new Point((e.getX())-((Frame.size.width-Screen.myWidth)/2),(e.getY())-((Frame.size.height-(Screen.myHeight))-(Frame.size.width-Screen.myWidth)/2));

}

}

import java.awt.\*;

public class Room {

public int worldWidth=12;

public int worldHeight=8;

public int blockSize=52;

public Block[][] block;

public Room(){

define();

}

public void define(){

block=new Block[worldHeight][worldWidth];

for (int y=0;y<block.length;y++)

{

for(int x=0;x<block[0].length;x++)

{

block[y][x]=new Block((Screen.myWidth/2)-((worldWidth\*blockSize)/2)+(x\*blockSize),y\*blockSize,blockSize, blockSize,Value.groundGrass,Value.airAir);

}

}

}

public void physic(){

for (int y=0;y<block.length;y++){

for(int x=0;x<block[0].length;x++){

block[y][x].physic();

}

}

}

public void draw(Graphics g)

{

for (int y=0;y<block.length;y++)

{

for(int x=0;x<block[0].length;x++)

{

block[y][x].draw(g);

}

}

for (int y=0;y<block.length;y++)

{

for(int x=0;x<block[0].length;x++)

{

block[y][x].fight(g);

}

}

}

}

import java.io.\*;

import java.util.\*;

public class Save {

public void loadSave(File loadPath)

{

try {

Scanner loadScanner=new Scanner(loadPath);

while(loadScanner.hasNext()){

Screen.killsToWin=loadScanner.nextInt();

for(int y=0;y<Screen.room.block.length;y++){

for(int x=0;x<Screen.room.block[0].length;x++){

Screen.room.block[y][x].groundID=loadScanner.nextInt();

}

}

for(int y=0;y<Screen.room.block.length;y++){

for(int x=0;x<Screen.room.block[0].length;x++){

Screen.room.block[y][x].airID=loadScanner.nextInt();

}

}

}

loadScanner.close();

} catch (Exception e) {

}

}

}

import java.awt.\*;

public class Store {

public static int shopWidth=8;

public static int buttonSize=52;

public static int cellSpace=2;

public static int awayFromRoom=29;

public static int iconSize=20;

public static int iconSpace=6;

public static int iconTextY=15;

public static int itemIn=4;

public static int heldID=-1;

public static int realID=-1;

public static int[] buttonID={Value.airTowerLaser,Value.airAir,Value.airAir,Value.airAir,Value.airAir,Value.airAir,Value.airAir,Value.airTrashCan};

public static int[] buttonPrice={10,0,0,0,0,0,0,0};

public Rectangle[] button=new Rectangle[shopWidth];

public Rectangle buttonHealth;

public Rectangle buttonCoins;

public boolean holdsItem=false;

public Store(){

define();

}

public void click(int mouseButton){

if(mouseButton==1){

for(int i=0;i<button.length;i++){

if(button[i].contains(Screen.mse)){

if(buttonID[i]!=Value.airAir){

if (buttonID[i]==Value.airTrashCan){

holdsItem=false;

}

else{

heldID=buttonID[i];

realID=i;

holdsItem=true;

}

}

}

}

if(holdsItem){

if(Screen.coinage>=buttonPrice[realID]){

for(int y=0;y<Screen.room.block.length;y++){

for(int x=0;x<Screen.room.block[0].length;x++){

if(Screen.room.block[y][x].contains(Screen.mse)){

if(Screen.room.block[y][x].groundID!=Value.groundRoad&&Screen.room.block[y][x].airID==Value.airAir){

Screen.room.block[y][x].airID=heldID;

Screen.coinage-=buttonPrice[realID];

}

}

}

}

}

}

}

}

public void define()

{

for(int i=0;i<button.length;i++)

{

button[i]=new Rectangle((Screen.myWidth/2)-((shopWidth\*(buttonSize+cellSpace))/2)+((buttonSize+cellSpace)\*i),(Screen.room.block[Screen.room.worldHeight-1][0].y)+Screen.room.blockSize+awayFromRoom,buttonSize,buttonSize);

}

buttonHealth=new Rectangle(Screen.room.block[0][0].x-1,button[0].y,iconSize,iconSize);

buttonCoins=new Rectangle(Screen.room.block[0][0].x-1,button[0].y+button[0].height-iconSize,iconSize,iconSize);

}

public void draw(Graphics g)

{

for(int i=0;i<button.length;i++){

if(button[i].contains(Screen.mse)){

g.setColor(new Color(255,255,255,100));

g.fillRect(button[i].x,button[i].y, button[i].width, button[i].height);

}

g.drawImage(Screen.tileset\_res[0],button[i].x,button[i].y, button[i].width, button[i].height,null);

if(buttonID[i]!=Value.airAir)

g.drawImage(Screen.tileset\_air[buttonID[i]], button[i].x+itemIn,button[i].y+itemIn, button[i].width-(itemIn\*2), button[i].height-(itemIn\*2),null);

if (buttonPrice[i]>0){

g.setColor(new Color(255,255,255));

g.setFont(new Font("Courier New",Font.BOLD,14));

g.drawString("$"+buttonPrice[i]+"",button[i].x+itemIn,button[i].y+itemIn+10);

}

}

g.drawImage(Screen.tileset\_res[1],buttonHealth.x, buttonHealth.y, buttonHealth.width, buttonHealth.height,null);

g.drawImage(Screen.tileset\_res[2],buttonCoins.x, buttonCoins.y, buttonCoins.width, buttonCoins.height,null);

g.setFont(new Font("Courier New",Font.BOLD,14));

g.setColor(new Color(255,255,255));

g.drawString(""+Screen.health,buttonHealth.x+buttonHealth.width+iconSpace,buttonHealth.y+iconTextY);

g.drawString(""+Screen.coinage,buttonCoins.x+buttonCoins.width+iconSpace,buttonCoins.y+iconTextY);

if (holdsItem){

g.drawImage(Screen.tileset\_air[heldID], Screen.mse.x-((button[0].width-(itemIn\*2) )/2)+itemIn, Screen.mse.y-((button[0].width-(itemIn\*2) )/2)+itemIn, button[0].width-(itemIn\*2), button[0].height-(itemIn\*2),null);

}

}

}

public class Value {

public static int groundGrass=0;

public static int groundRoad=1;

public static int airAir=-1;

public static int airCave=0;

public static int mobGreeny=0;

public static int mobAir=-1;

public static int airTrashCan=1;

public static int airTowerLaser=2;

public static int[] deathReward={5};

}

# ПРИЛОЖЕНИЕ Б

Текстуры, разработанные для программы

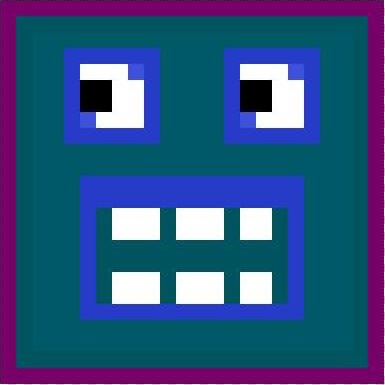


Рисунок 1 - Юнит.

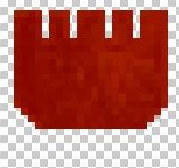


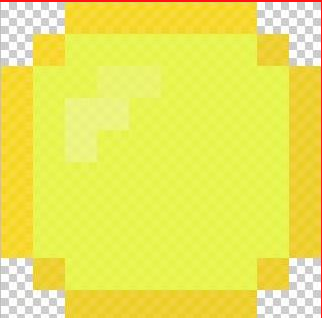
Рисунок 2 - Башня.



Рисунок 3 - Отмена выбора башни.



Рисунок 4 - Выход. Точка назначения для юнитов.

  
Рисунок 5 - Золото.

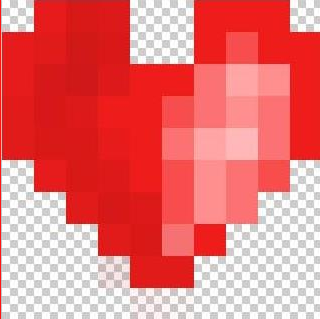
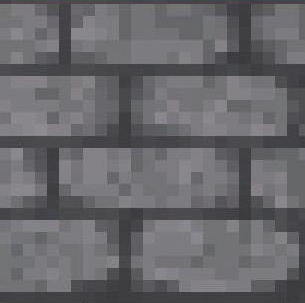


Рисунок 6 - Здоровье.



Рисунок 7 - Трава.

  
Рисунок 8 - Дорога.