МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики и технологий программирования

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

«2D платформер на Unity»

Выполнил:

студент гр.653502 Болох И.П.

Проверил: Козуб В.Н.

г. Минск

2017 год

**Содержание**

**ВВЕДЕНИЕ 2**

**ЦЕЛИ И ЗАДАЧИ 3**

**1 ТЕОРЕТИЧСКАЯ ЧАСТЬ 4**

1.1 2D ИЗМЕРЕНИЕ, ТИПЫ ГРАФИКИ 4

1.2 ПРАВИЛА ПЛАТФОРМЕРОВ 5

**2 РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА 7**

2.1 ПРАВИЛА ИГРЫ 7

2.2 РАЗРАБОТА ПРОГРАММЫ 8

2.3 ДАЛЬНЕЙШИЕ ПЛАНЫ НА ДОРАБОТКУ ИГРЫ 14

**3 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 15**

**ЗАКЛЮЧЕНИЕ 16**

**СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 17**

**1**

**Введение**

Ещё до освоения технологии 3D игры широко укоренились, как развлечение или хобби среди разных слоев населения. Развитие видео-игр пришло на конец 20 века и активно продолжается в наши дни, раньше игры были двумерными, по причине технических ограничений техники и малой осведомленности в области гейм-дизайна. Потому гейм-дизайнерам приходилось очень много работать и находить новые решения ,чтобы заинтересовать пользователя, благодаря чему были открыты многочисленные новые жанры видео-игр. Среди них: аркады, платформеры, скороллеры, симуляторы, стратегии, головоломки и множество других.

Игры в жанре “Платформер” являются одним из старейших жанров и за многие годы развития выработались определенные правила и схемы, по которым эти игры создаются и по сей день. Однако основная концепция спустя годы осталась неизменной.

Платформер— жанр компьютерных игр, в которых основной чертой игрового процесса является прыгание по платформам, лазанье по лестницам, собирание предметов, обычно необходимых для завершения уровня. Наиболее известными представителями данного жанра являются:Soldat, Contra, Metal Slug, Metroid.

Unity – широко известная бесплатная интегрированная среда для разработки игр и приложений в 2D и 3D измерениях. Unity поддерживает широкое множество платформ, таких как Windows, iOS, Linux, Android, Xbox One, PS Vita, PS 4.

Игра “Pocan” выполнена на Unity. Язык C# был выбран по причине своей обильной структуры и широкого функционала, что позволяет воплотить в реальность любую задумку гейм-дизайнера.

2

ЦЕЛИ И ЗАДАЧИ

Основной целью данной курсовой работы является изучение и создание 2D платформера на основе интегрированной среды разработки Unity, а также программирование компонентов на языке C#.

Данная работа состоит из 3 разделов: первый раздел теоретический, содержащий общие сведения по теме, второй раздел практический, где описывается код и идея игры, а так же дальнейшие планы на разработку. Третий раздел – руководство пользователя.

Основные задачи, выполнение для написания работы:

1) Изучение необходимой литературы

2) Изучение методов для создания ИИ, управления, интерфейса, действий персонажа

3) Изучение жанра платформеров

4) Изучение правил платформеров

6) Написание кода программы

7) Создание игры жанра платформер

8) Анализ полученной игры

3

1. **Теоретический раздел**
   1. **2D ИЗМЕРЕНИЕ, ТИПЫ ГРАФИКИ**

С приходом новых технологий и развитием графики появилась возможность делать игры 3D, однако и привычное многим 2D не потеряло своей актуальности и в наши дни. Благодаря таланту современных гейм-дизайнеров и художников 2D игры так же могут радовать нас прекрасными видами и архитектурой уровней, примером тому может

служить недавняя Ori and the Blind Forest.

Графика 2D это плоскость с двумя координатными осями x, y, без понятия глубины. В 2D используются спрайты.

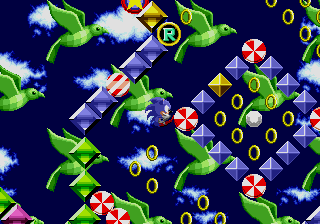
Однако, не всё так просто, в разных играх используются разные типы 2D графики, среди них выделяются:Low-poly, пиксельная или рисованная

4

графика. В 90-е годы была популярна фотореалистичная графика, такая использовалась в Mortal kombat.

В игре “Pocan” использовалась 2D-растровая графика, благодаря которой был достигнут эффект яркости, “сказочности” и красочности происходящего.

* 1. **ПРАВИЛА ПЛАТФОРМЕРОВ**

 Основной чертой игрового процесса является прыгание по платформам, лазанье по лестницам, собирание предметов, обычно необходимых для завершения уровня. Некоторые предметы, называемые пауэр-апами (англ. power-up), наделяют управляемого игроком персонажа особой силой, которая обычно иссякает со временем (к примеру: силовое поле, ускорение, увеличение высоты прыжков). Коллекционные предметы, оружие и «пауэер-ап» собираются обычно простым прикосновением персонажа и для применения не требуют специальных действий со стороны игрока. Реже предметы собираются в «инвентарь» героя и применяются специальной командой (такое поведение более характерно для аркадных головоломок). Сходный жанр компьютерных игр SideScroller.

Противники (называемые «врагами»), всегда многочисленные и разнородные, обладают примитивным искусственным интеллектом, стремясь максимально приблизиться к игроку, либо не обладают им 5

вовсе, перемещаясь по круговой дистанции или совершая повторяющиеся действия. Соприкосновение с противником обычно отнимает жизненные силы у героя или вовсе убивает его. Иногда противник может быть нейтрализован либо прыжком ему на голову, либо из оружия, если им обладает герой. Смерть живых существ обычно изображается упрощённо или символически (существо исчезает или проваливается вниз за пределы экрана).



Уровни, как правило, изобилуют секретами (скрытые проходы в стенах, высокие или труднодоступные места), нахождение которых существенно облегчает прохождение и подогревает интерес игрока.

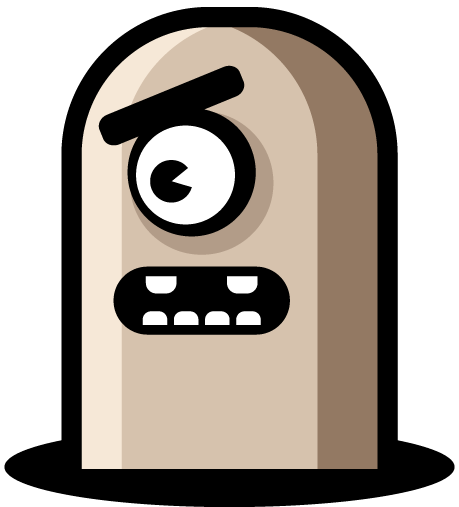
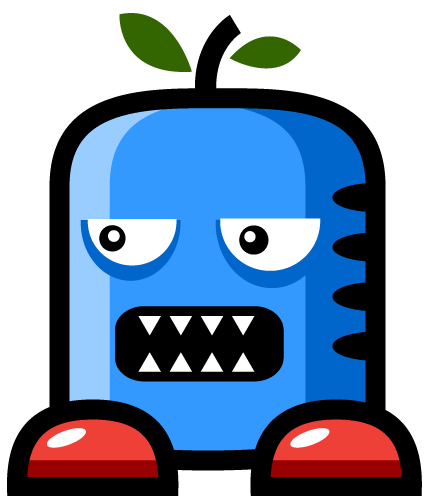
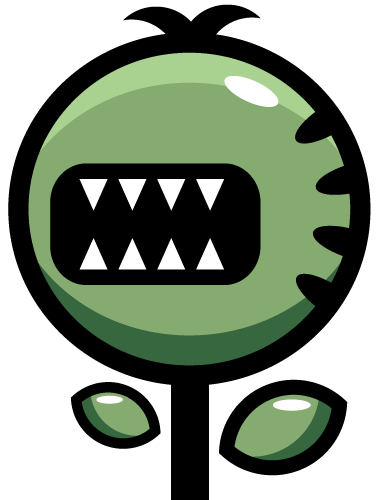
Игры подобного жанра характеризуются нереалистичностью, рисованной мультяшной графикой. Героями таких игр обычно бывают мифические существа (к примеру: драконы, гоблины) или антропоморфные животные.

6

**2 РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА**

**2.1 ПРАВИЛА ИГРЫ**

Игрок появляется на уровне и должен уничтожить всех противников волшебными снарядами по нажатию клавиши lctrl. Персонаж(рис.1) имеет 5 очков здоровья. Для перехода наследующий уровень необходимо уничтожить всех противников.



(рис.1)(рис.2) (рис.3)(рис.4)

Подробнее о противниках. Растение(рис.2)стоит на месте и стреляет в одну сторону, убивается при прыжке на него или при попадании магией.

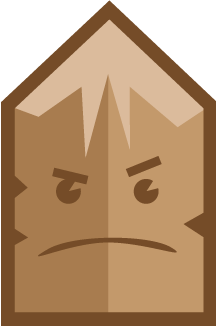
Ходячий монст(рис.3) передвигается и наносит урон при прикосновении, убивается при прыжке на него.

Червь(рис.4) стоит на месте и наносит урон при прикосновении, убивается при попадании магией или от прыжка на него.

Игровые обьекты.

Деревянный кол(рис.5) наносит урон при прыжке на него, не может быть уничтожен.

Сердце(рис.6) добавляет здоровье при поднятии.



(рис.5) (рис.6)

7

**2.2 РАЗРАБОТА ПРОГРАММЫ**

Далее будут продемонстрированы наиболее важные части кода программы.

Камера в игре привязана к передвижению персонажа и управляется на AD,Space.

using UnityEngine;  
using System.Collections;  
  
public class CameraController : MonoBehaviour  
{  
    [SerializeField]  
    private float speed = 2.0F;  
  
    [SerializeField]  
    private Transform target;  
  
    private void Awake()  
    {  
        if (!target) target = FindObjectOfType<Character>().transform;  
    }  
  
    private void Update()  
    {  
        Vector3 position = target.position;         position.z = -10.0F;  
        transform.position = Vector3.Lerp(transform.position, position, speed \* Time.deltaTime);  
    }  
}

Привязка камеры к персонажу <Character>осуществляется if (!target) target = FindObjectOfType<Character>().transform;

Для плавного движения камеры в след за персонажем

transform.position = Vector3.Lerp(transform.position, position, speed \* Time.deltaTime);

Сам персонаж:

using UnityEngine;  
using System.Collections;  
  
public class Character : Unit  
{  
    [SerializeField]  
    private int lives = 5;\\кол-во жизней 8  
  
    public int Lives  
    {  
        get { return lives; }  
        set  
        {  
           if (value < 5) lives = value;  
            livesBar.Refresh();  
        }  
    }  
    private LivesBar livesBar;  
  
    [SerializeField]  
    private float speed = 3.0F; [\\ скорость](file:///\\скорость) передвижения  
    [SerializeField]  
    private float jumpForce = 15.0F; [\\высота](file:///\\высота) прыжка  
  
    private bool isGrounded = false;  
  
    private Bullet bullet;  
  
    private CharState State  
    {  
        get { return (CharState)animator.GetInteger("State"); }  
        set { animator.SetInteger("State", (int)value); }  
    }  
  
    new private Rigidbody2D rigidbody;  
    private Animator animator;  
    private SpriteRenderer sprite;  
  
    private void Awake()  
    {  
        livesBar = FindObjectOfType<LivesBar>();  
        rigidbody = GetComponent<Rigidbody2D>();  
        animator = GetComponent<Animator>();  
        sprite = GetComponentInChildren<SpriteRenderer>();  
  
        bullet = Resources.Load<Bullet>("Bullet");   
    }  
  
    private void FixedUpdate()  
    {  
        CheckGround();  
    }  
  
    private void Update()  
    { 9  
        if (isGrounded) State = CharState.Idle;  
  
        if (Input.GetButtonDown("Fire1")) Shoot();\\для стрельбы на lctrl или ЛКМ  
        if (Input.GetButton("Horizontal")) Run(); [\\управление](file:///\\управление) на AD  
        if (isGrounded && Input.GetButtonDown("Jump")) Jump(); [\\привязка](file:///\\привязка) прыжка на Space  
    }  
  
    private void Run()  
    {  
        Vector3 direction = transform.right \* Input.GetAxis("Horizontal");  
  
        transform.position = Vector3.MoveTowards(transform.position, transform.position + direction, speed \* Time.deltaTime);  
  
        sprite.flipX = direction.x < 0.0F;  
  
        if (isGrounded) State = CharState.Run;  
    }  
  
    private void Jump()  
    {  
        rigidbody.AddForce(transform.up \* jumpForce, ForceMode2D.Impulse);  
    }  
  
    private void Shoot()  
    {  
        Vector3 position = transform.position; position.y += 0.8F;  
        Bullet newBullet = Instantiate(bullet, position, bullet.transform.rotation) as Bullet;  
  
        newBullet.Parent = gameObject;  
        newBullet.Direction = newBullet.transform.right \* (sprite.flipX ? -1.0F : 1.0F);  
    }  
  
    public override void ReceiveDamage()  
    {  
        Lives--;  
  
        rigidbody.velocity = Vector3.zero;  
        rigidbody.AddForce(transform.up \* 8.0F, ForceMode2D.Impulse);  
  
        Debug.Log(lives);  
    } 10  
  
    private void CheckGround()  
    {  
        Collider2D[] colliders = Physics2D.OverlapCircleAll(transform.position, 0.3F);  
  
        isGrounded = colliders.Length > 1;  
  
        if (!isGrounded) State = CharState.Jump;  
    }  
  
    private void OnTriggerEnter2D(Collider2D collider)  
    {  
  
        Bullet bullet = collider.gameObject.GetComponent<Bullet>();  
        if (bullet && bullet.Parent != gameObject)  
        {  
            ReceiveDamage(); [\\уменьшение](file:///\\уменьшение) здоровья при попадании по персонажу  
        }  
    }  
}  
  
  
public enum CharState  
{  
    Idle,  
    Run,  
    Jump  
}

ИИ для Червя

using UnityEngine;  
using System.Collections;  
  
public class Monster : Unit  
{  
    protected virtual void Awake() { }  
    protected virtual void Start() { }  
    protected virtual void Update() { }  
  
    protected virtual void OnTriggerEnter2D(Collider2D collider)  
    {  
        Bullet bullet = collider.GetComponent<Bullet>();  
  
        if (bullet)  
        {  
            ReceiveDamage(); [\\смерть](file:///\\смерть) червя при попадании по нему 11  
        }  
  
        Character character = collider.GetComponent<Character>();  
  
        if (character)  
        {  
            character.ReceiveDamage();\\урон персонажу при попадании  
        }  
    }  
}

ИИ для Ходячего монстра

using UnityEngine;  
using System.Collections;  
  
public class Monster : Unit  
{  
    protected virtual void Awake() { }  
    protected virtual void Start() { }  
    protected virtual void Update() { }  
  
    protected virtual void OnTriggerEnter2D(Collider2D collider)  
    {  
        Bullet bullet = collider.GetComponent<Bullet>();  
  
        if (bullet)  
        {  
            ReceiveDamage();\\урон монстру при попадании по нему магией  
        }  
  
        Character character = collider.GetComponent<Character>();  
  
        if (character)  
        {  
            character.ReceiveDamage();\\урон персонажу при столкновении  
        }  
    }  
}

ИИ для растения

using UnityEngine;  
using System.Collections;  
  
public class ShootableMonster : Monster  
{  
    [SerializeField]  
    private float rate = 2.0F; 12  
    [SerializeField]  
    private Color bulletColor = Color.white;\\выбор цвета пули (По умолчанию белый)  
  
    private Bullet bullet;  
  
    protected override void Awake()  
    {  
        bullet = Resources.Load<Bullet>("Bullet");  
    }  
  
    protected override void Start()  
    {  
        InvokeRepeating("Shoot", rate, rate);\\постоянная стрельба растения  
    }  
  
    private void Shoot()  
    {  
        Vector3 position = transform.position;

 position.y += 0.5F;  
        Bullet newBullet = Instantiate(bullet, position, bullet.transform.rotation) as Bullet;  
  
        newBullet.Parent = gameObject;  
        newBullet.Direction = -newBullet.transform.right;  
        newBullet.Color = bulletColor;  
    }  
  
    protected override void OnTriggerEnter2D(Collider2D collider)[\\защита](file:///\\защита) от самоубийства монстра  
    {  
        Unit unit = collider.GetComponent<Unit>();  
  
        if (unit && unit is Character)[\\убийство](file:///\\убийство) растения при прыжке на него  
        {  
            if (Mathf.Abs(unit.transform.position.x - transform.position.x) < 0.3F) ReceiveDamage();  
            else unit.ReceiveDamage();  
        }  
    }  
}

13

**2.3 ДАЛЬНЕЙШИЕ ПЛАНЫ НА ДОРАБОТКУ ИГРЫ**

- Необходимо реализовать эффект от полной потери здоровья

- Эффект от поднятия сердец

-Игровое меню

-Звуки

-4-6 игровых уровней и переход между ними

14

**3.РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

Разрабатываемый проект – игра под названием ”Pocan”.

Для использования продукта необходимы:

-Операционная система Windows

-Клавиатура, мышь

- 520 мб свободного места на жестком диске

Запуск игры осуществляется по запуску файла Pocan.exe далее вы увидите заставку и перейдете к игровому процессу.

Игровой процесс крайне прост, для передвижения персонажа в горизонтальной плоскости используйте клавиши A,D

Для прыжков используйте клавишу Space

Для ведения огня по противникам используйте ЛКМ или left control

15

**ЗАКЛЮЧЕНИЕ**

В результате проведенной работы была создана компьютерная игра “Pocan” и документация к ней.

В процессе работы над программой были изучены литературные и интернет источники, развиты навыки работы с компонентами среды Visual Studio 2015, языка программирования С#, а так же Unity и ее возможностей при создании компьютерных игр.

Несмотря на незавершенность, игровой пользователь уже может получить представление об игровом процессе и о жанре в целом.

16

**СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

Проектирование и архитектура игр **- Э.Роллингз, Д.Моррис**

**Интернет источники:**

[**https://mva.microsoft.com/ru/training-courses/-unity-8635?l=Cv4CmYF1\_6304984382**](https://mva.microsoft.com/ru/training-courses/-unity-8635?l=Cv4CmYF1_6304984382)

[**https://www.youtube.com/channel/UC1QkejUSZjFfNhoMjISVJMA**](https://www.youtube.com/channel/UC1QkejUSZjFfNhoMjISVJMA)

[**https://itvdn.com/ru/blog/article/creating-super-meat-boy-with-unity**](https://itvdn.com/ru/blog/article/creating-super-meat-boy-with-unity)

**bevoulin.com/category/game-asset**

[**https://www.youtube.com/channel/UCzxRv9BtqrM946JmaMLtv\_w**](https://www.youtube.com/channel/UCzxRv9BtqrM946JmaMLtv_w)

[**https://ru.wikipedia.org/wiki/Платформер**](https://ru.wikipedia.org/wiki/Платформер)

**17**

14