Муниципальное автономное общеобразовательное учреждение г.Владимира «Средняя общеобразовательная школа № 36»

Проект по информатике

По теме: Разработка игры "SUPER-MEGA DESTROYER"

в среде Unity 3D

Выполнил:

Ученик 11 «А» класса

Забалуев Владислав

Научные руководители:

Кузнецов Д.А.,

Абрахин С.И.

1

Оглавление

1. Введение……………………………………………………....3
2. Обзор методов…………………………………………....…..4

2.1 Краткое описание задачи……………………………......4

2.2 Описание метода 1………………………………………4

2.3 Описание метода 2………………………………………5

1. Разработка программы……………………………………...8

3.1 Сюжет……………………………..…………………..…8

3.2 Алгоритм работы программы…………………………..8

3.2.1 Главный герой………………………………………..8

3.2.2 Боты…………………………………………………...9

3.2.3 Колышки……………………………………………..10

3.2.4 «Гаврики» и классы………………………………….10

3.3 Описание интерфейса пользователя…………………..10

3.3.1 Главное меню………………………………………..10

3.3.2 Меню паузы…………………………………………11

3.3.3 Дополнения…………………………………………12

1. Результаты тестирования……………………………………13

5. Выводы………………………………………………………19

6. Источники информации……………………………………19

7. Приложение…………………………………………………20

7.1-7.16 Коды………………………………………………20-39

Введение

Среди различных предложенных тем проектов я выбрал реализацию компьютерной игры. В будущем мне хотелось бы связать свою жизнь с игровой индустрией, возможно, работать в большой команде программистов в знаменитой на весь мир корпорации и поражать всё игровое сообщество новенькой игрой. Но сейчас, когда мой путь в программирование только начинается, я решил сделать относительно простой 2D платформер.

Меня вдохновили на это такие игры как : «Valiant Hearts: The Great War», «Cuphead» и «Ori and The Blind Forest». Каждая из них будет по-своему интересна игроку и, несомненно, завлечет его в своей увлекательный мир.

3

Обзор методов

2.1 Краткое описание задачи:

Этот проект представляет собой полноценный 2D платформер.

В нем главный герой умеет бегать, прыгать, стрелять, цепляться за стены.

Враги же должны патрулировать определенную область; при приближении игрока бежать за ним, пытаясь убить; если игрок убежит на определенное расстояние возвращаться на патрулирование.

У всех юнитов имеется определенное количество жизней, если оно равно 0, то юнит умирает, и анимация.

Существует препятствие, коснувшись которого главный получит урон

Также в игре имеется обычное меню, меню паузы, ползунок, регулирующий громкость, информация об игре (управление и цель).

Стоит сказать, что все рисунки и анимации я сделал сам с помощью Adobe Illustrator и Spriter.

2.2 Обзор метода 1

Сейчас, когда существует несколько десятков (если не сотен) движков для игр, у начинающего программиста был огромный просто в выборе. Я решил присмотреться к двум бесплатным самым популярным на текущий момент игровым движкам. Первым изучаемым стал **Unreal Engine** — игровой движок, разрабатываемый и поддерживаемый компанией Epic Games. Первой игрой на этом движке был шутер от первого лица Unreal, выпущенный в 1998 году. Хотя движок первоначально был предназначен для разработки шутеров от первого лица, его последующие версии успешно применялись в играх самых различных жанров, в том числе стелс-играх, файтингах и массовых многопользовательских ролевых онлайн-играх. Написанный на языке C++, движок позволяет создавать игры для большинства операционных систем и платформ: Microsoft

4

Windows, Linux, Mac OS и Mac OS X; консолей Xbox, Xbox 360, Xbox One, PlayStation 2, PlayStation 3, PlayStation 4, PSP, PS Vita, Wii, Dreamcast, GameCube и др., а также на различных портативных устройствах, например, устройствах Apple (iPad, iPhone), управляемых системой iOS и прочих. (Впервые работа с iOS была представлена в 2009 году[4], в 2010 году продемонстрирована работа движка на устройстве с системой webOS[5]).

Для упрощения портирования движок использует модульную систему зависимых компонентов; поддерживает различные системы рендеринга (Direct3D, OpenGL, Pixomatic; в ранних версиях: Glide, S3, PowerVR), воспроизведения звука (EAX, OpenAL, DirectSound3D; ранее: A3D), средства голосового воспроизведения текста, распознавание речи, модули для работы с сетью и поддержки различных устройств ввода.

Для игры по сети поддерживаются технологии Windows Live, Xbox Live, GameSpy и прочие, включая до 64 игроков (клиентов) одновременно. Таким образом, движок адаптировали и для применения в играх жанра MMORPG (один из примеров: Lineage II).

2.3 Обзор метода 2

**Unity** — межплатформенная среда разработки компьютерных игр, разработанная американской компанией Unity Technologies. Unity позволяет создавать приложения, работающие на более чем 25 различных платформах, включающих персональные компьютеры, игровые консоли, мобильные устройства, интернет-приложения и другие. Выпуск Unity

состоялся в 2005 году и с того времени идёт постоянное развитие.

5

На Unity написаны тысячи игр, приложений, визуализации математических моделей, которые охватывают множество платформ и жанров. При этом Unity используется как крупными разработчиками, так и независимыми студиями.

Как правило, игровой движок предоставляет множество функциональных возможностей, позволяющих их задействовать в различных играх, в которые входят моделирование физических сред, карты нормалей, динамические тени и многое другое. В отличие от многих игровых движков, у Unity имеется два основных преимущества: наличие визуальной среды разработки и межплатформенная поддержка. Первый фактор включает не только инструментарий визуального моделирования, но и интегрированную среду, цепочку сборки, что направлено на повышение производительности разработчиков, в частности, этапов создания прототипов и тестирования. Под межплатформенной поддержкой предоставляется не только места развертывания (установка на персональном компьютере, на мобильном устройстве, консоли и т. д.), но и наличие инструментария разработки (интегрированная среда может использоваться под Windows и Mac OS).

Третьим преимуществом называется модульная система компонентов Unity, с помощью которой происходит конструирование игровых объектов, когда последние представляют собой комбинируемые пакеты функциональных элементов. В отличие от механизмов наследования, объекты в Unity создаются посредством объединения функциональных блоков, а не помещения в узлы дерева наследования. Такой подход облегчает создание прототипов, что актуально при разработке игр.

В качестве недостатков приводятся ограничение визуального редактора при

6

работе с многокомпонентными схемами, когда в сложных сценах визуальная работа затрудняется. Вторым недостатком называется отсутствие поддержки Unity ссылок на внешние библиотеки, работу с которыми программистам приходится настраивать самостоятельно, и это также затрудняет командную работу. Ещё один недостаток связан с использованием шаблонов экземпляров (англ. prefabs). С одной стороны, эта концепция Unity предлагает гибкий подход визуального редактирования объектов, но с другой стороны, редактирование таких шаблонов является сложным. Также, WebGL-версия движка, в силу специфики своей архитектуры (трансляция кода из C# в С++ и далее в JavaScript), имеет ряд нерешённых проблем с производительностью, потреблением памяти и работоспособностью на мобильных устройствах.

Ещё один недостаток — это потеря точности float в компоненте Transform по всем трем осям. Начинается это проявляться на координатах 10000 и больше, после этой координаты начинаются артефакты.

2.4 Обоснование выбора

В итоге я всё-таки выбрал Unity. Мой выбор связан с прочтением множества статей, связанных со сравнением движков. В них пишут, что Unity является наиболее простым для начинающего программиста, по нему больше русскоязычный информации в интернете, да и вообще «он не так плох».

Но стоит сказать, что для реализации простого 2D платформера подойдет практически любой движок, а различия Unity и Unreal Engine больше выделяются в больших проектах(таких, например, как Ori and The Blind Forest). Кстати, чаще всего у крупных студий имеется свой собственный движок.

7

Разработка программы

3.1 Сюжет

При создании игры меня гораздо больше техническая и графическая составляющая, нежели сюжетная. Задумкой игры стал парнишка, с автоматом, желающий добраться до своего любимого объекта (им стал обычный кустик) через толпы монстров и кучи преград.

3.2 Алгоритм работы программы:

3.2.1 Главный герой

У главного героя при изменении координаты по Х проигрывается анимация ходьбы. Это проверяется нажатием клавиш «A» и «D», стрелочек.

Главный герой умеет прыгать, при этом у него проигрывается анимация прыжка. Персонажу дается своего рода пинок, в результате которого он подлетает вверх.

Чтобы герою нельзя было давать не ограниченное количество пинков, существует проверка «Checkground». Она проверяет находиться ли в области героя земля, или иной объект, от которого можно оттолкнуться, если да, он есть, тогда можно прыгать, если его нет, то нельзя.

Герой умеет поворачиваться в сторону ходьбы. В этом ему помогает метод Flip. В нем считывается какая нажата клавиша. Если А или стрелочка влево, то герой поворачивается именно влево, а если D или стрелка вправо, то герой поворачивается вправо.

Герой умеет стрелять. Для этого я создал 2 отдельных скрипта (1 для пули 1 для самого оружия). В скрипте пули программа считывает: нажата ли «Fire1» (кнопка F). Если да, то вызывается метод, создающий пулю. Пуля же просто летит по направлению, в которое смотрел герой при выстреле, и

8

уничтожается при столкновении или пролетев по времени 1,5 фрейма. Также у героя есть определенная точка из которой он ведет стрельбу(FirePoint). У ботов её нет.

Также у героя имеются жизни, которые отображаются на экране. При попадании вражеской пули, столкновением с врагом, колышком их количество уменьшается, и это отображается как пустое сердце на месте заполненного. Получая очередной урон герой аж подпрыгивает от радости. Все это получается при проверке касания игроком чего-либо с тегом «pipka» или можно прописать это отдельным кодом (код колышка).

Если количество жизней у героя меньше 1, тогда уровень перезапускается.

3.2.2 Боты

Боты по своей сути умеют все то же самое, что и герой, кроме прыжка.

У каждой маленькой машинки для убийства есть своя точка, вокруг которой он ходит в состоянии «Chill», но стоит игроку приблизиться так наш бот уже бежит стрелять в маленького беззащитного ниндзю с автоматом, переходя в состояние «Angry». Но если герой убежит на достаточное расстояние до бот переходит в состояние «goBack» и бежит обратно к своей точке, где снова включается состояние «Chill».

У бота в разном состоянии разное направление поворота, но суть его остается такой же, как и у героя.

Стрельба у бота зависит от координаты героя. Пуля летит в то направление, где находиться герой. Это реализуется сравнением координат бота и героя. Также стрельба осуществляется с небольшой автоматически просчитываемой задержкой.

Анимация у ботов 1-бег, так как он постоянно бегает.

9

3.2.3 Колышки

Если колышка касается юнит и этот юнит-главный герой, тогда наносить урон.

3.2.4 Гаврики и классы

Назвать гавриков, кроме как «гаврики» я не могу. Они просто стоят слева от места респауна и смотрят в сторону героя. Также их можно убить, так как они имеют скрипт монстра, в котором сказано, что при попадании пули, боту должен нанестись урон. А в скрипте класса юнит происходит уменьшение количества жизней у персонажей, также прописана их смерть.

3.3 Описание интерфейса пользователя

3.3.1 Главное меню



Такое жизнерадостное меню встречает нас при первом запуске.

10

Кнопа «Start» своим действием загрузит нас на первый уровень.

Кнопка «Information about game» подразумевает по собой открытие нового окна(не сцены) , где игрок может узнать цель игры(когда печатал текст,я напечатал его дважды и не заметил, в связи с этим цель игры упоминается дважды) и управление.



Кнопка «Exit» выходит из игры. Причем что в обычном меню, что в меню паузы.

Кнопка «Return» возвращает игрока в главное меню.

3.3.2 Меню паузы

Зайдя в него, время в игре останавливается.

11



Кнопка «Restart» перезапускает уровень.

Кнопка «Resume» продолжает игру.

Кнопка «Menu» выходит в главное меню.

Кнопка «Exit», как было упомянуто ранее-выход из игры.

Ползунок позволяет регулировать громкость музыки на уровне (после смерти или рестарта) сбрасывается.

3.3.3 Дополнения

Также существует сцена, для игрока, который прошел игру, но я её прикреплять не буду. Пусть она останется загадкой.

Вся игра сопровождается музыкой. В главном меню 1 песня, в самой игре другая. Особая музыка ждет тех, кто прошел игру.

Также в игре присутствует паралакс заднего фона.

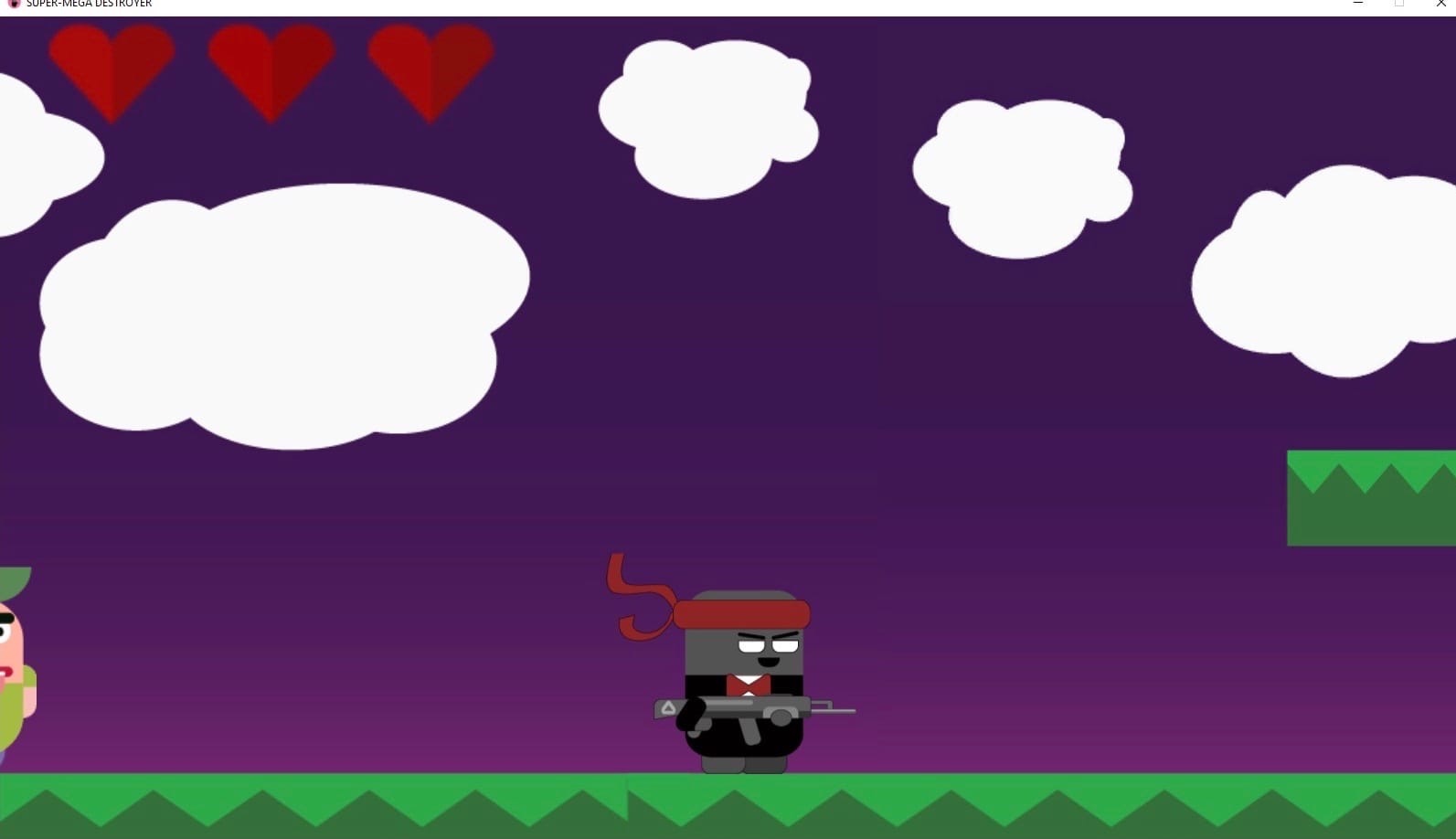
12

Результаты тестирования

Как я уже говорил, зайдя в игру нас встречает такое главное меню.

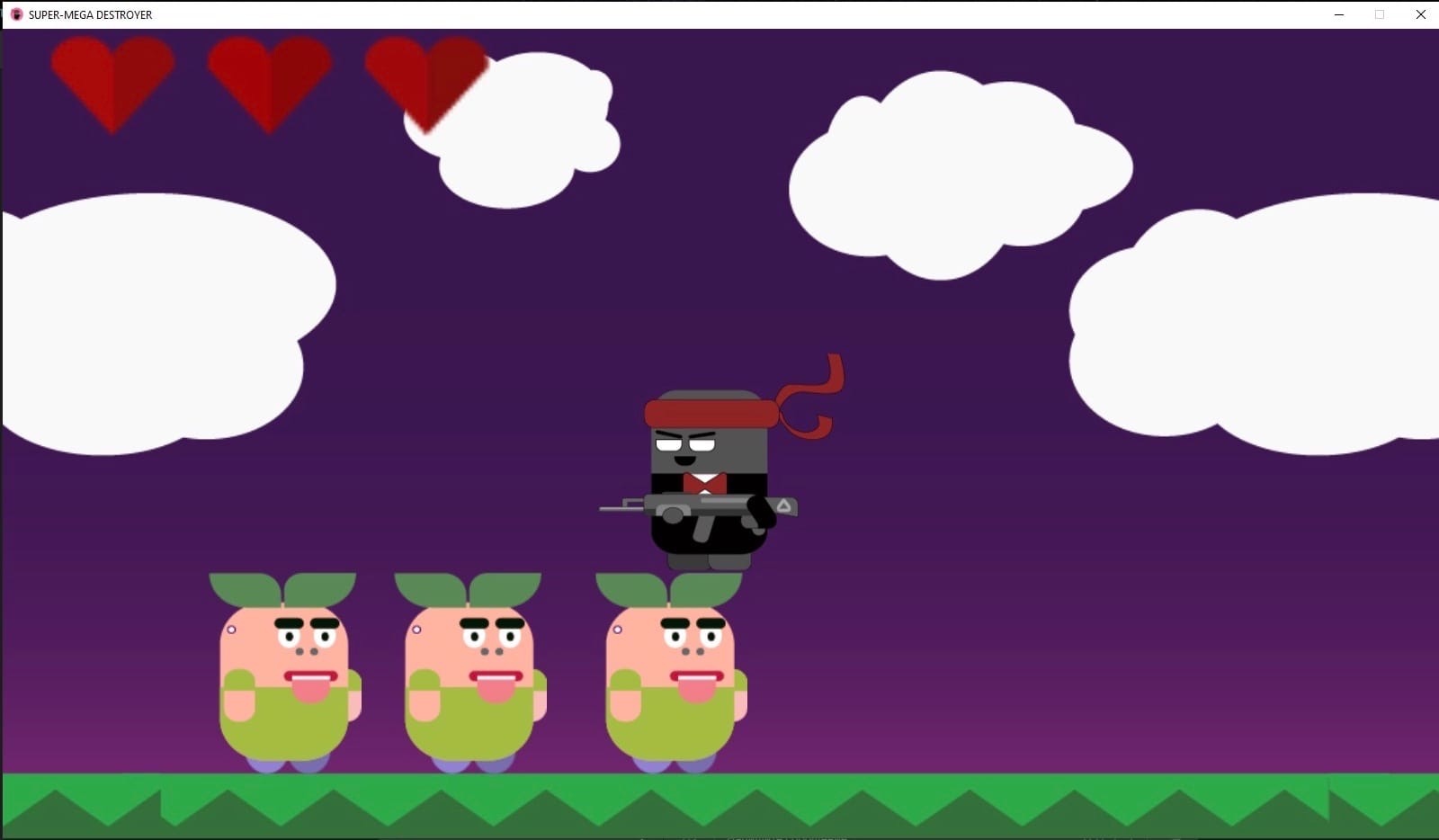


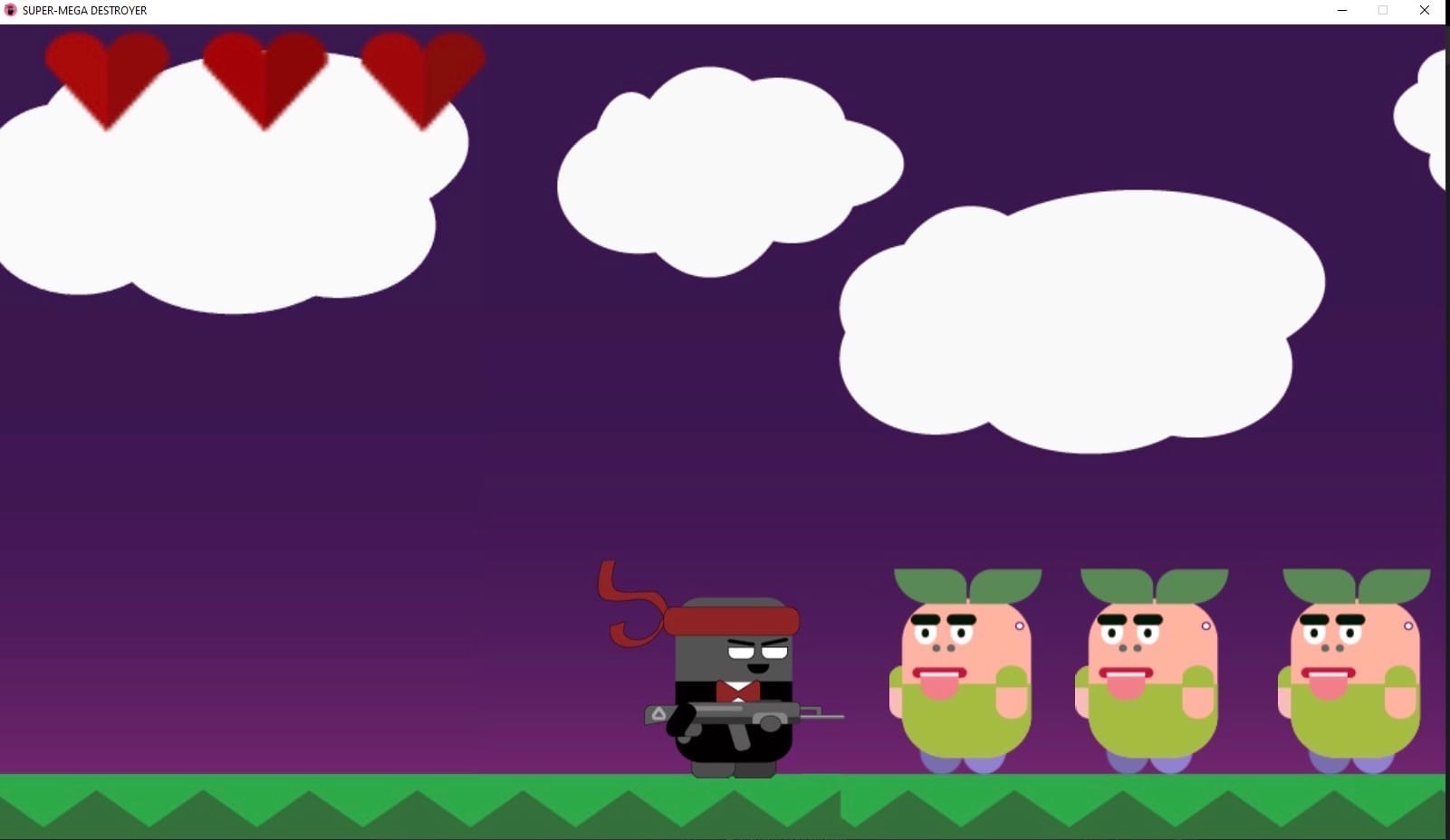
Нажав кнопку «Srart», мы переместимся на сцену с самой игрой.



13

Слева от нас находятся «гаврики», которые смотрят в сторону героя

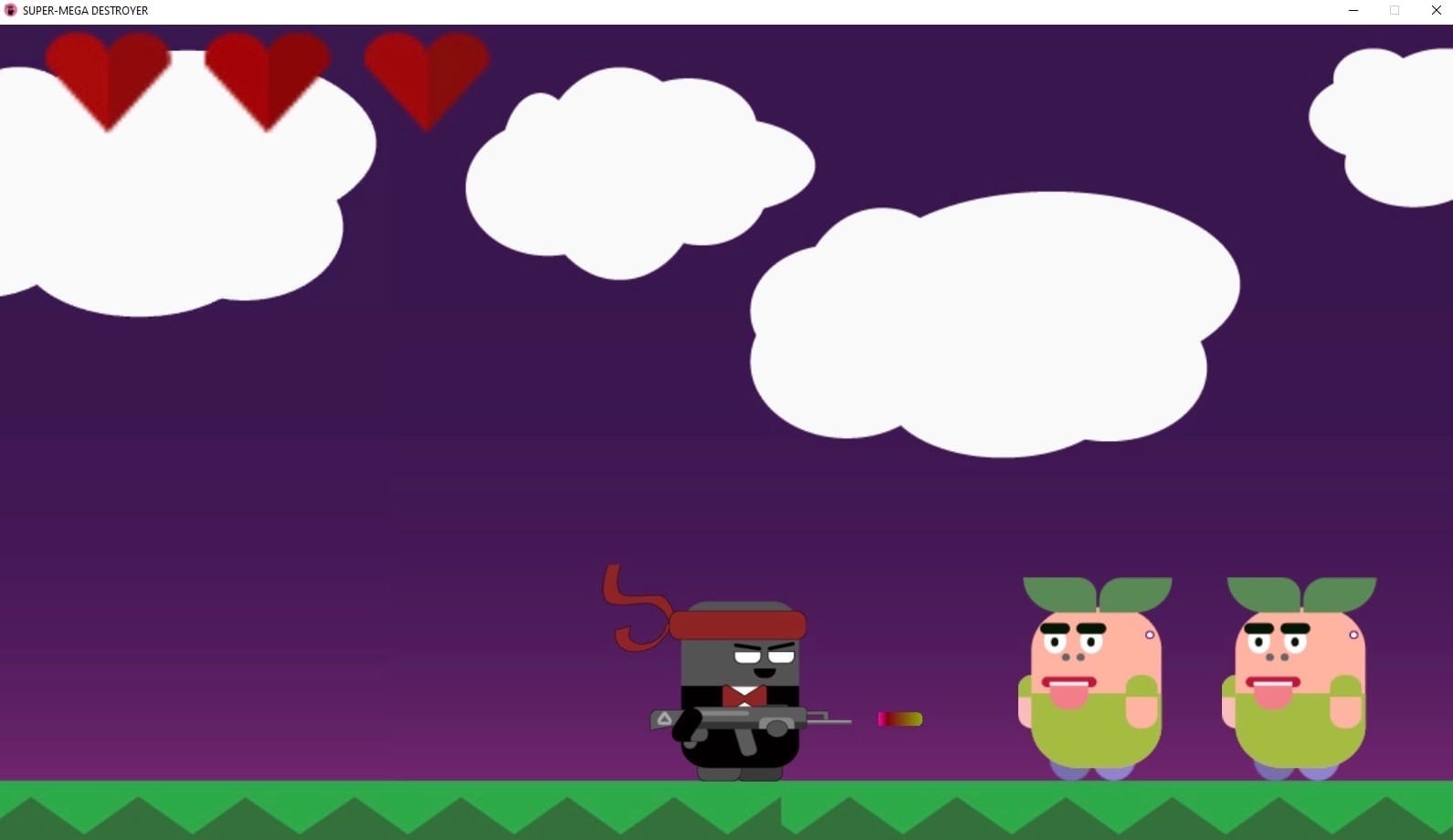




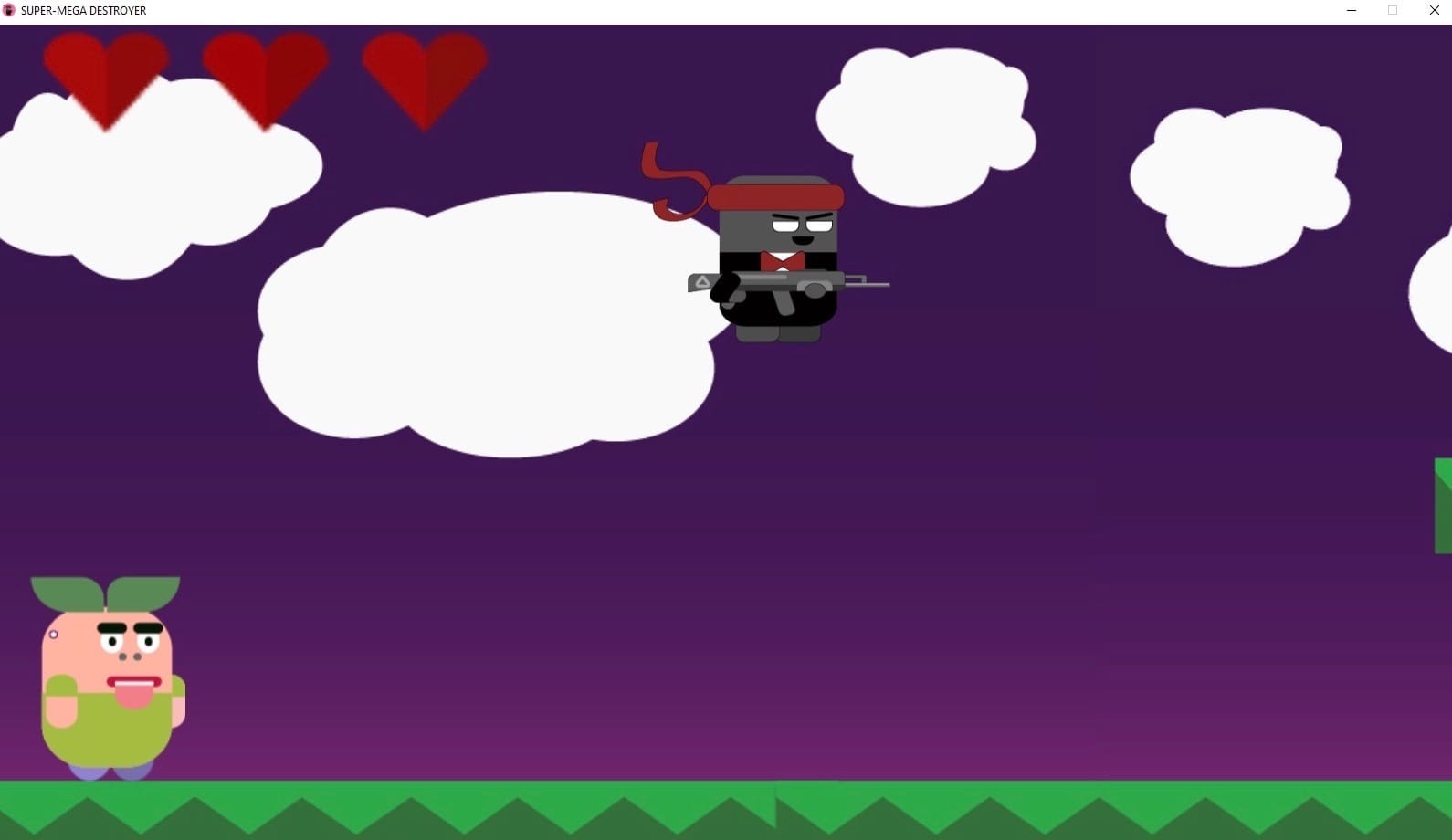
Наш герой переместился, и они теперь смотрят в другую сторону.

14

Главный герой умеет стрелять, а «гаврики» умирать.

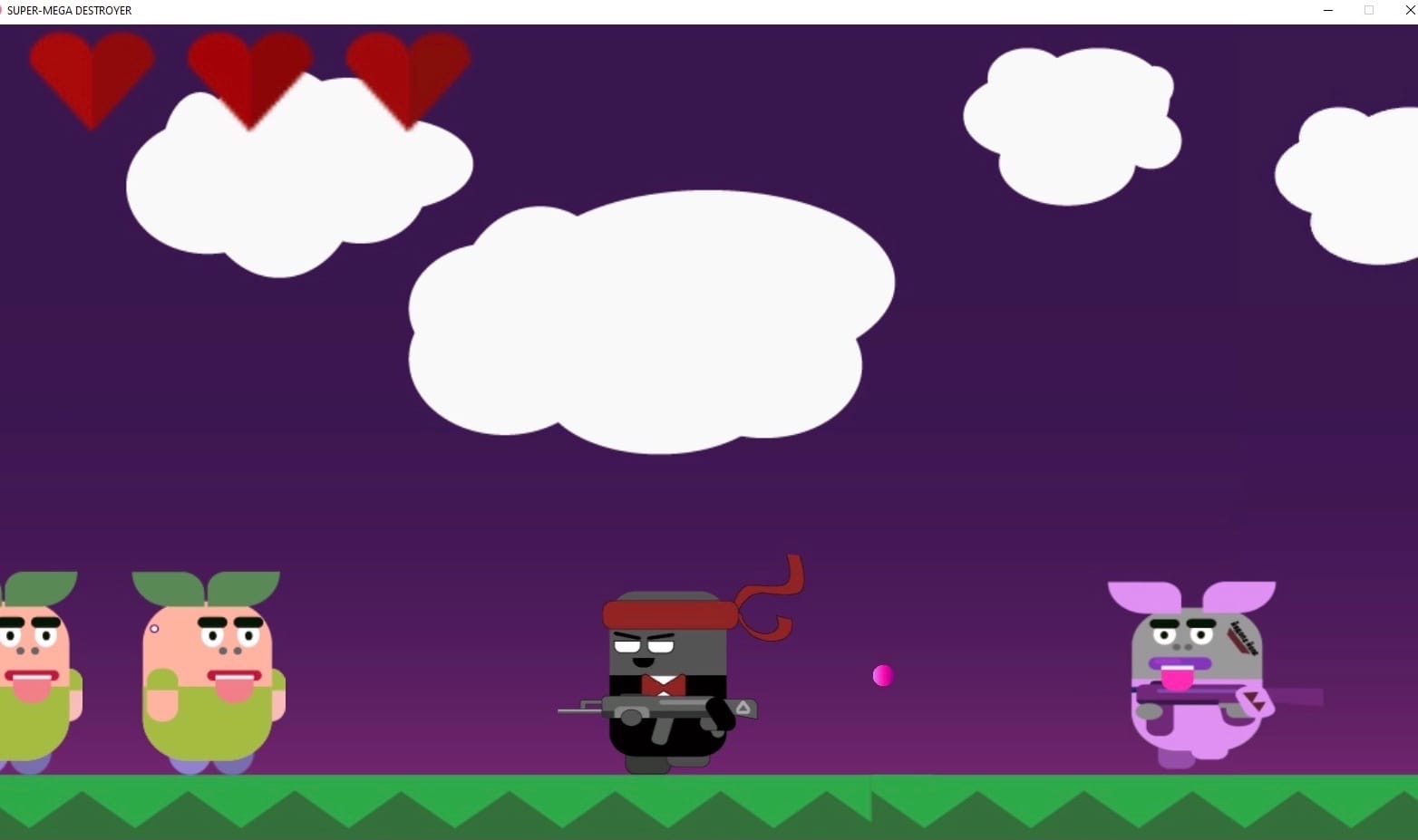


Прыжок главного героя

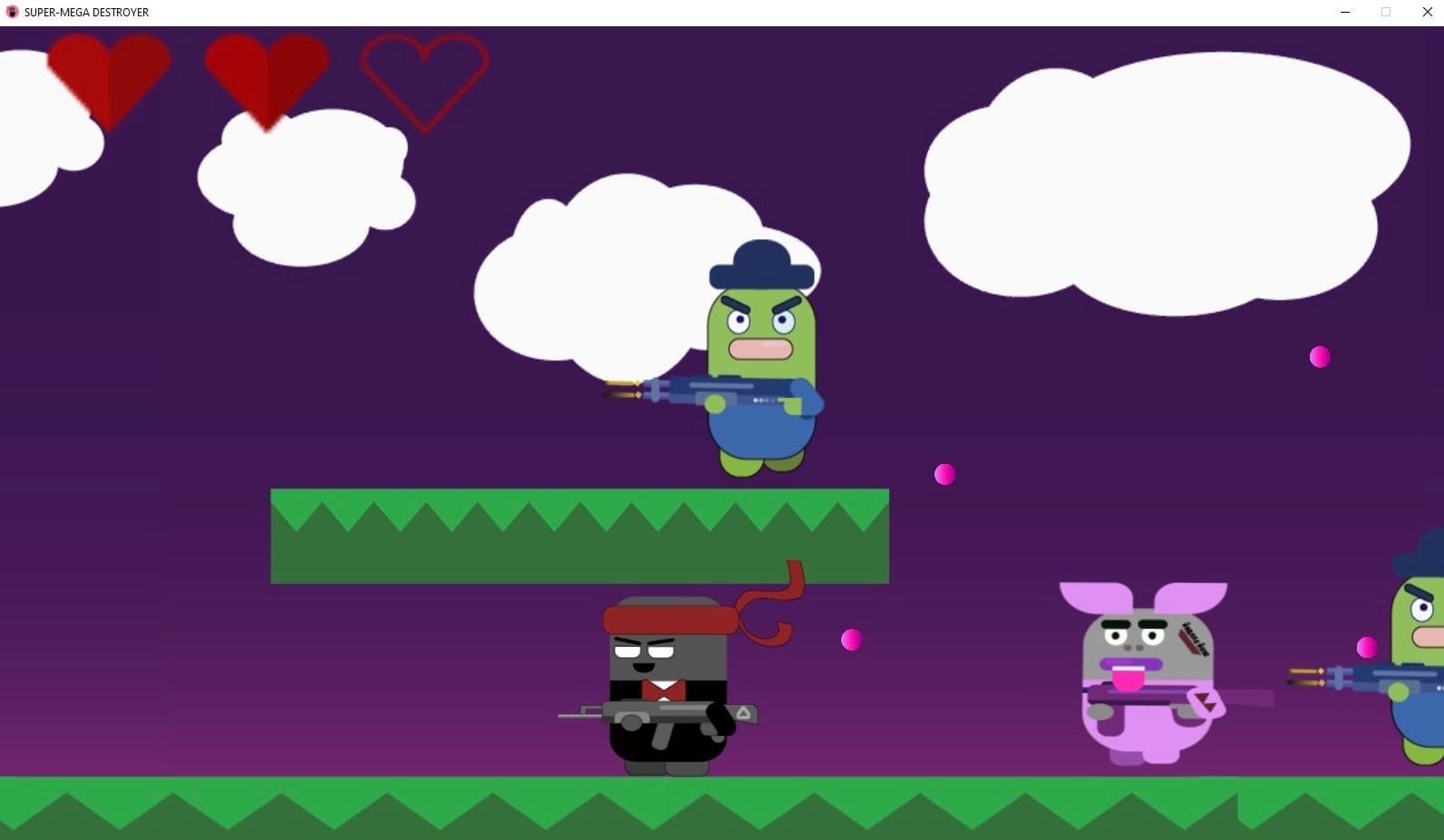


15

Стреляющий в главного героя злодей1.

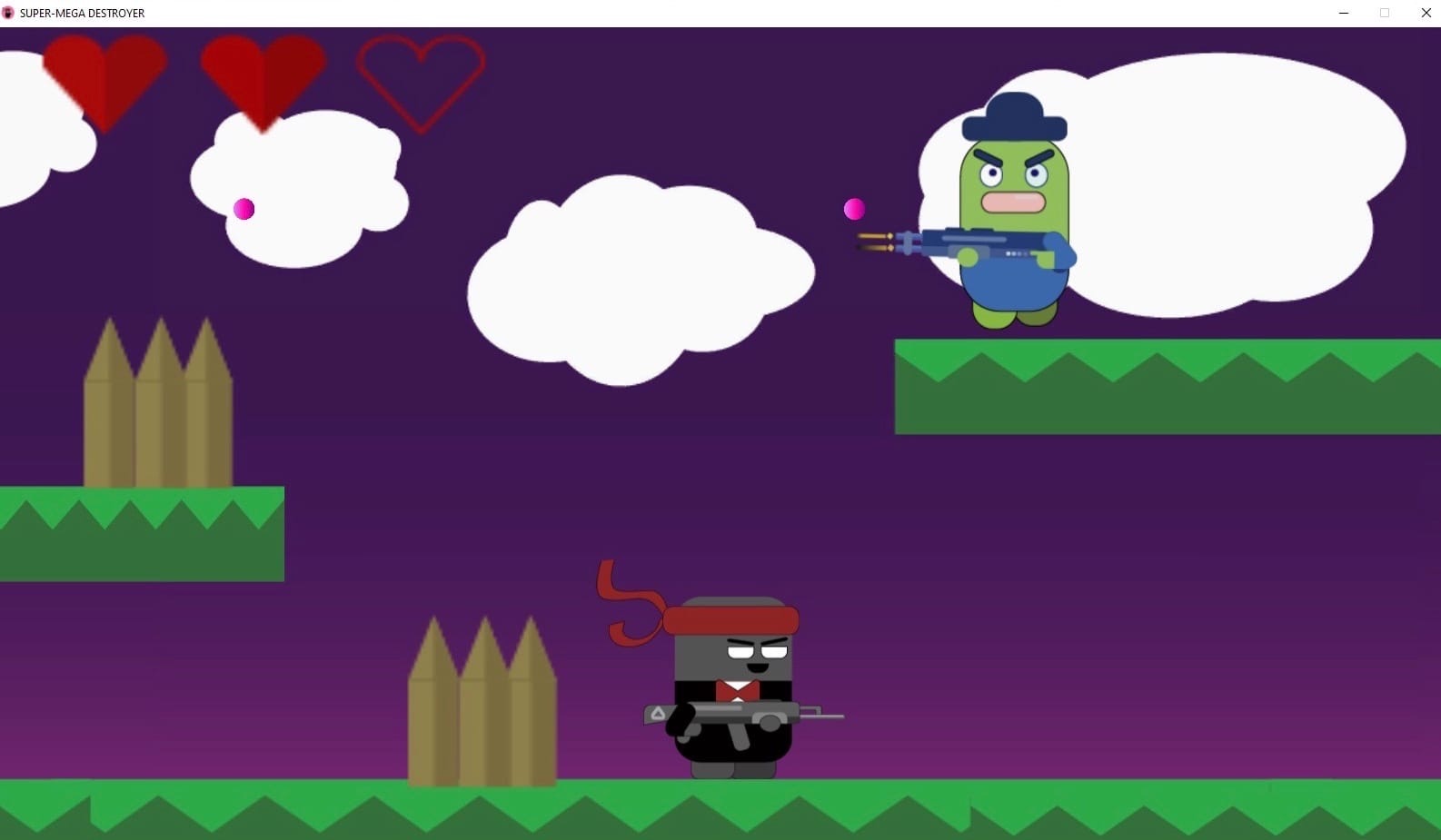


Стреляющие, бегающие к главному герою враги; уменьшение количества жизней у героя отображается на экране.



16

Колья, наносящие урон герою.



Патрулирование злодеем определенной области с поворотом в сторону ходьбы





17

Смещение заднего фона и земли при прохождение героем определенного расстояние в ту или иную сторону



(последние скриншоты были сделаны в Юнити, так как игроку это никога не увидеть)

При нажатии «Esc» игра ставиться на паузу. Функции меню паузы описаны выше.



18

Выводы

Мне кажется, я справился с поставленной задачей, так как мой проект вполне можно считать самостоятельной игрой. Наибольшую сложность составило программирование самого персонажа(особенно стрельба(я посмотрел десяток видео, переделывая все с самого на чала по не сколько раз) , так как я начал именно с этого. Сейчас, когда мой проект уже готов, я могу сказать, что создавать игры (даже самые простые) довольно сложно, но интересно. Я не жалею ни о минуте, проведенной за рисованием персонажа, настройкой юнити или подбором музыки.

Источники информации

К сожалению, ссылки на форумы, на которых я сидел по началу, найти теперь не представляется возможным. Но основным источником информации стали :

<https://www.youtube.com/channel/UCy8aONm_W21gtK2D8KKnINA> – Flatingo. Канал разработчика, делающего инди-игры. На нем можно найти многое про игровую индустрию, научиться азам, найти мотивацию делать игры и посмотреть, как это получается у других + именно тут выбрал, в каком стиле будет моя игра.

[https://www.youtube.com/watch?v=CAPVBTKk3Ww&lis t=PL9OfdQ50vg-PNtEZLtk21LmfM1ZrmV9iw&index=3&t=985s](https://www.youtube.com/watch?v=CAPVBTKk3Ww&lis%20t=PL9OfdQ50vg-PNtEZLtk21LmfM1ZrmV9iw&index=3&t=985s) - отличное видео, позволяющее сделать простенький платформер, но , к сожалению, у меня не получилось сделать по этому видео стрельбу, да и вообще оно учит лишь самым основам.

<https://www.youtube.com/channel/UCHr72IH-33mrD2h09MR3N3Q> - Graphical Design. По мне, лучший канал, на котором реально учат делать игры с 0 и до конца. Человек, делающий для него видео помог мне советом (что говорит о взаимосвязи с аудиторией)

19

Приложение

7.1 Код главного героя

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.SceneManagement;

public class mainhero : Unit

{

[SerializeField]

private bool isGrounded = false;

public int health;

public int numberOfLives;

public Image[] lives;

public Sprite fullLive;

public Sprite emptyLive;

Rigidbody2D rb;

private Animator anim;

private SpriteRenderer sprite;

private void Awake()

{

rb = GetComponent<Rigidbody2D>();

anim = GetComponent<Animator>();

sprite = GetComponentInChildren<SpriteRenderer>();

}

private void FixedUpdate()

{

rb.velocity = new Vector2(Input.GetAxis("Horizontal") \* 7f, rb.velocity.y);

Checkground();

Flip();

}

void Update()

{

if (isGrounded && Input.GetKeyDown(KeyCode.Space))

{

jump();

anim.SetTrigger("Jump");

}

if (isGrounded && Input.GetAxis("Horizontal") == 0)

{

20

anim.SetBool("isRunnig", false);

}

if (isGrounded && Input.GetAxis("Horizontal") != 0)

{

Flip();

anim.SetBool("isRunnig", true);

}

if (health <1)

{

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 0);

}

}

void Flip()

{

if (Input.GetAxis("Horizontal") < 0)

transform.localRotation = Quaternion.Euler(0, 180, 0);

if (Input.GetAxis("Horizontal") > 0)

transform.localRotation = Quaternion.Euler(0, 0, 0);

}

// переворот

void jump()

{

rb.AddForce(transform.up \* 14f, ForceMode2D.Impulse);

}

// прыжок

private void Checkground()

{

Collider2D[] colliders = Physics2D.OverlapCircleAll(transform.position, 2F);

isGrounded = colliders.Length > 1;

}

// проверка на наличие земли

private void OnCollisionEnter2D(Collision2D helplz)

{

if (helplz.gameObject.tag == "pipka")

{

health--;

}

if (helplz.gameObject.tag == "Finish")

{

SceneManager.LoadScene("winMenu");

}

}

void Reloadlevel()

{

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 0);

}

public override void ReceiveDamage()

{

health--;

rb.velocity = Vector3.zero;

rb.AddForce (transform.up \* 6f, ForceMode2D.Impulse);

// Debug.Log(health);

21

if (health > numberOfLives)

{

health = numberOfLives;

}

for (int i = 0; i < lives.Length; i++)

{

if (i < health)

{

lives[i].sprite = fullLive;

}

else

{

lives[i].sprite = emptyLive;

}

if (i < numberOfLives)

{

lives[i].enabled = true;

}

else

{

lives[i].enabled = false;

}

}

}

}

// урон

22

7.2 Код оружия игрока

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Weapon : MonoBehaviour

{

public Transform firePoint;

public GameObject bullet;

public float shootCooldown=0.25f ;

private void Start()

{

shootCooldown = 1f;

}

// Update is called once per frame

void Update()

{

if (Input.GetButtonDown("Fire1"))

{

Shoot();

}

if (shootCooldown > 0)

{

shootCooldown -= Time.deltaTime;

}

}

void Shoot()

{

Instantiate(bullet, firePoint.position, firePoint.rotation);

}

}

7.3 Код пули игрока

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Bullet : MonoBehaviour

{

public float speed = 10f;

public Rigidbody2D rb;

void Start()

{

rb.velocity = transform.right \* speed;

Destroy(gameObject, 1.5F);

}

private void OnTriggerEnter2D(Collider2D hitInfo)

{

// Debug.Log(hitInfo.name);

Destroy(gameObject);

}

}

23

7.4 Код Unit

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Unit : MonoBehaviour

{

public int Hitpoint=5;

public virtual void ReceiveDamage()

{

Hitpoint--;

Debug.Log(Hitpoint);

if (Hitpoint == 0)

{

Die();

}

}

protected virtual void Die()

{

Destroy(gameObject);

}

}

7.5 Код Монстров(общий)

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Monster : Unit

{

private void OnTriggerEnter2D(Collider2D collider)

{

Bullet bullet = collider.GetComponent<Bullet>();

if (bullet)

{

ReceiveDamage();

}

}

}

7.6 Код колышка

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Obstacle : MonoBehaviour

{

private void OnTriggerEnter2D(Collider2D collider)

{

Unit unit = collider.GetComponent<Unit>();

24

if (unit && unit is mainhero )

{

unit.ReceiveDamage();

}

}

}

7.7 Код «Гаврика»

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class EnemyFollow : MonoBehaviour

{

public float speed=5f;

public Transform target;

// public GameObject player;

void Start()

{

target = GameObject.FindGameObjectWithTag("Player").GetComponent<Transform>();

// player = GameObject.FindGameObjectWithTag("Player");

}

// Update is called once per frame

void Update()

{

/\* if (Vector2.Distance(transform.position,target.position)>3)

{

transform.position = Vector2.MoveTowards(transform.position, target.position, speed \* Time.deltaTime);

} \*/

if (transform.position.x> target.position.x)

{

transform.localRotation = Quaternion.Euler(0, 180, 0);

}

if (transform.position.x < target.position.x)

{

transform.localRotation = Quaternion.Euler(0, 0, 0);

}

}

}

Здесь скажу, что не стал удалять и «гаврика» код ходьбы.

Изначально он просто бегал за героем.

7.8 Код Злодеев

using System.Collections;

using System.Collections.Generic;

using System.Security.Cryptography.X509Certificates;

using UnityEngine;

public class Enemy : MonoBehaviour

25

{

public float speed;

public int positionOfPatrol;

public Transform point;

public float stoppingdistance;

private float timeBTWshots;

public float starttimeBTWshots;

public GameObject projectile;

private Transform player;

private Animator anim;

bool moveingRight;

bool chill = false;

bool angry = false;

bool goBack = false;

void Start()

{

player = GameObject.FindGameObjectWithTag("Player").transform;

timeBTWshots = starttimeBTWshots;

anim = GetComponent<Animator>();

}

void Update()

{

if (Vector2.Distance(transform.position, point.position) < positionOfPatrol && angry==false)

{

chill=true;

}

if (Vector2.Distance(transform.position, player.position) < stoppingdistance)

{

angry = true;

chill = false;

goBack = false;

}

if (Vector2.Distance(transform.position, player.position) > stoppingdistance)

{

goBack = true;

angry = false;

}

if (chill == true)

{

Chill();

}

else if (angry == true)

{

Angry();

}

26

else if (goBack == true)

{

GoBack();

}

}

void Chill()

{

if (transform.position.x > point.position.x + positionOfPatrol)

{

moveingRight = false;

transform.localRotation = Quaternion.Euler(0, 0, 0);

}

if (transform.position.x < point.position.x - positionOfPatrol)

{

moveingRight =true;

transform.localRotation = Quaternion.Euler(0, 180, 0);

}

if (moveingRight)

{

transform.position = new Vector2(transform.position.x + speed \* Time.deltaTime, transform.position.y);

}

else

{

transform.position = new Vector2(transform.position.x - speed \* Time.deltaTime, transform.position.y);

}

}

void Angry()

{

transform.position = Vector2.MoveTowards(transform.position, player.position, speed \* Time.deltaTime);

if (transform.position.x > player.position.x)

{

transform.localRotation = Quaternion.Euler(0, 0, 0);//поворот за героем

if (timeBTWshots <= 0)// задержка в стрельбе,сама стрельба

{

Instantiate(projectile, transform.position, Quaternion.identity);

timeBTWshots = starttimeBTWshots;

}

else

{

timeBTWshots -= Time.deltaTime;

}

}

if (transform.position.x < player.position.x)

{

transform.localRotation = Quaternion.Euler(0, 180, 0);

if (timeBTWshots <= 0)

{

Instantiate(projectile, transform.position, Quaternion.identity);

27

timeBTWshots = starttimeBTWshots;

}

else

{

timeBTWshots -= Time.deltaTime;

}

}

}

void GoBack()

{

transform.position = Vector2.MoveTowards(transform.position, point.position, speed \* Time.deltaTime);

if (transform.position.x < point.position.x-positionOfPatrol)

{

transform.localRotation = Quaternion.Euler(0, 180, 0);

}

if (transform.position.x > point.position.x+positionOfPatrol)

{

transform.localRotation = Quaternion.Euler(0, 0, 0);

}

}

}

7.9 Код пули злодея

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Projectile : MonoBehaviour

{

public float speed=27f;

Rigidbody2D rb;

private Transform player;

private Vector2 target;

void Start()

{

player = GameObject.FindGameObjectWithTag("Player").transform;

target = new Vector2(player.position.x, player.position.y);

rb = GetComponent<Rigidbody2D>();

if (transform.position.x > player.position.x)

{

rb.velocity = new Vector2(-speed, rb.velocity.y);

}

if (transform.position.x < player.position.x)

{

rb.velocity = new Vector2(speed, rb.velocity.y);

28

}

Destroy(gameObject, 2F);

}

void Update()

{

if (transform.position.x == target.x && transform.position.y== target.y)

{

DestroyProjectile();

}

}

private void OnTriggerEnter2D(Collider2D other)

{

if (other.CompareTag("Player"))

{

DestroyProjectile();

}

Unit unit = other.GetComponent<Unit>();

if (unit && unit is mainhero)

{

unit.ReceiveDamage();

}

}

void DestroyProjectile()

{

Destroy(gameObject);

}

}

7.10 Код главного меню

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Main\_Menu : MonoBehaviour

{

public void PlayGame()

{

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);

}

public void QuitGame()

{

Debug.Log("Quit");

Application.Quit();

}

}

29

7.11 Код меню паузы

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class Pause\_menu : MonoBehaviour

{

public static bool GameIsPaused = false;

public GameObject pauseMenuUI;

// Update is called once per frame

void Update()

{

if (Input.GetKeyDown(KeyCode.Escape))

{

if (GameIsPaused)

{

Resume();

}

else

{

Pause();

}

}

}

public void Resume()

{

pauseMenuUI.SetActive(false);

Time.timeScale = 1f;

GameIsPaused = false;

}

void Pause()

{

pauseMenuUI.SetActive(true);

Time.timeScale = 0f;

GameIsPaused = true;

}

public void LoadMenu()

{

SceneManager.LoadScene("Menu");

Time.timeScale = 1f;

}

public void QuitGame()

{

Debug.Log("quit");

Application.Quit();

}

}

7.12 Код рестарта

using System.Collections;

using System.Collections.Generic;

30

using UnityEngine;

using UnityEngine.SceneManagement;

public class Restart : MonoBehaviour

{

public void RestartLevel()

{

SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 0);

Time.timeScale = 1f;

}

}

7.13 Код камера

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class CameraControler : MonoBehaviour

{

[SerializeField]

private float speed = 6.0F;

[SerializeField]

private Transform target;

private void Awake()

{

if (!target) target = FindObjectOfType<mainhero>().transform;

}

private void Update()

{

Vector3 position = target.position; position.y = 0.1f; position.z = -8.0F;

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

}

}

7.14 Код ползунка звука

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class VolumeValue : MonoBehaviour

{

private AudioSource audioSrc;

private float musicVolume=0.05f;

void Start()

{

audioSrc = GetComponent<AudioSource>();

}

31

void Update()

{

audioSrc.volume = musicVolume;

}

public void SetVolume(float vol)

{

musicVolume = vol;

}

}

7.15 Код паралакса

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Scrolling : MonoBehaviour

{

public bool scrolling, paralax;

public float backgroundSize;

public float paralaxSpeed;

private Transform cameraTransform;

private Transform[] layers;

private float viewZone=15f;

private int leftIndex;

private int rightIndex;

private float lastcameraX;

private void Start()

{

cameraTransform = Camera.main.transform;

layers = new Transform[transform.childCount];

lastcameraX = cameraTransform.position.x;

for (int i=0; i<transform.childCount; i++)

{

layers[i] = transform.GetChild(i);

}

leftIndex = 0;

rightIndex = layers.Length-1;

}

private void Update()

{

if (paralax)

{

float deltaX = cameraTransform.position.x - lastcameraX;

transform.position += Vector3.right \* (deltaX \* paralaxSpeed);

}

lastcameraX = cameraTransform.position.x;

32

if (scrolling)

{

if (cameraTransform.position.x < (layers[leftIndex].transform.position.x + viewZone))

{

ScrollLeft();

}

if (cameraTransform.position.x > (layers[rightIndex].transform.position.x - viewZone))

{

ScrollRight();

}

}

}

private void ScrollLeft()

{

int lastRight = rightIndex;

layers[rightIndex].position = Vector3.right \* (layers[leftIndex].position.x - backgroundSize);

leftIndex = rightIndex;

rightIndex--;

if (rightIndex < 0)

{

rightIndex = layers.Length - 1;

}

}

private void ScrollRight()

{

int lastLeft = leftIndex;

layers[leftIndex].position = Vector3.right \* (layers[rightIndex].position.x + backgroundSize);

rightIndex = leftIndex;

leftIndex++;

if (leftIndex == layers.Length)

{

leftIndex = 0;

}

}

}

7.16 Скрипт экрана победы

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class winner : MonoBehaviour

{

public void LoadMenu()

{

SceneManager.LoadScene("Menu");

}

}

33