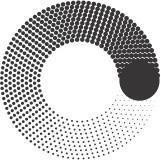
**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ВЫСШАЯ ШКОЛА ПЕЧАТИ И МЕДИАИНДУСТРИИ**

***Институт Принтмедиа и информационных технологий Кафедра Информатики и информационных технологий***

**направление подготовки**

**09.03.02 «Информационные системы и технологии»**

**ЛАБОРАТОРНАЯ РАБОТА № 1**

**Дисциплина:**

**Тема:**

**Выполнил(а): студент(ка) группы**

**Ву Хоанг Нам Куан 211-728**

(Фамилия И.О.)

**Дата, подпись *Quân\_***

(Дата) (Подпись)

**Проверил**

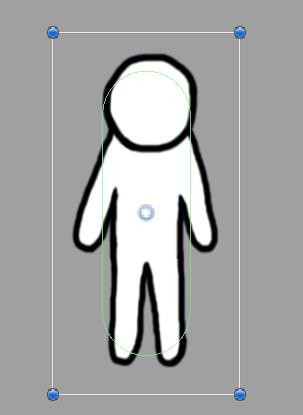
(Фамилия И.О., степень, звание) **(Оценка)**

**Дата, подпись**

(Дата) (Подпись)

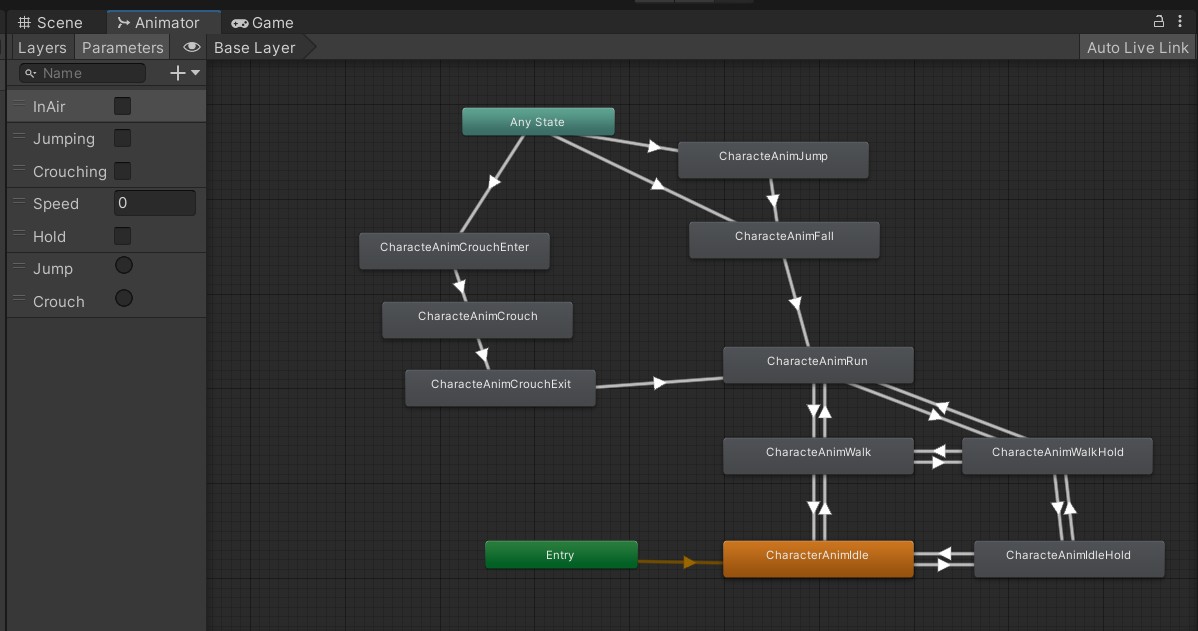
**Замечания:**

# Главный герой:



Основные персонажи включают: Sprite Renderer, Animator, Rigidbody2D, Capsule Collider 2D, Player Control Script.

Animator:



«Аниматор» для управления состояниями персонажа, используйте параметры для переключения между состояниями, различные анимации. Бегать, прыгать, сидеть, стоять на месте и т.д....

Script:

Скрипт управления персонажем содержит более 500 строк, поэтому я не могу скопировать и вставить его сюда.

Сценарий имеет очень быстрый итеративный цикл, именно благодаря которому игра работает, этот цикл будет проверять состояние персонажа, получать вводы с клавиатуры, мыши и... и выполнять функции доступного кода. Я добавил несколько вещей, таких как: HP, скорость бега, ускорение бега, ускорение остановки, сила прыжка, двойной прыжок, гравитация, направления движения с помощью клавиш на клавиатуре, ...

//Handle phys cs vo d F xedUpdate()

{

f ( s\_dead) return;

f ( sCl mb ng)

{

}

else

{

}

r g d.grav tyScale = 0f;

r g d.veloc ty = new Vector2(r g d.veloc ty.x, vert cal \* speed); return;

r g d.grav tyScale = normalGrav ty;

//Movement veloc ty

float des redSpeed = Mathf.Abs(move\_ nput.x) > 0.1f o move\_ nput.x \* move\_max : 0f; float accelerat on = Mathf.Abs(move\_ nput.x) > 0.1f o move\_accel : move\_deccel; accelerat on = ! s\_grounded o jump\_move\_percent \* accelerat on : accelerat on;

move.x = Mathf.MoveTowards(move.x, des redSpeed, accelerat on \* T me.f xedDeltaT me);

UpdateFac ng(); UpdateJump(); UpdateCrouch();

//Move

r g d.veloc ty = move;

}

//Handle render and controls vo d Update()

{

f ( s\_dead)

return;

h t\_t mer += T me.deltaT me; grounded\_t mer += T me.deltaT me;

vert cal = Input.GetAx sRaw("Vert cal"); f ( sLadder && Mathf.Abs(vert cal) > 0f)

{

sCl mb ng = true;

}

f ( sIns deSp kes)

{

TakeDamage(10);

}

//Controls

fllayerControls controls = fllayerControls.Get(player\_ d);

move\_ nput = !d sable\_controls o controls.GetMove() : Vector2.zero; jump\_press = !d sable\_controls o controls.GetJumpDown() : false; jump\_hold = !d sable\_controls o controls.GetJumpHold() : false;

f (jump\_press || move\_ nput.y > 0.5f) Jump();

//Reset when fall

f (transform.pos t on.y < fall\_pos\_y - GetS ze().y)

{

TakeDamage(max\_hp \* fall\_damage\_percent); f (reset\_when\_fall)

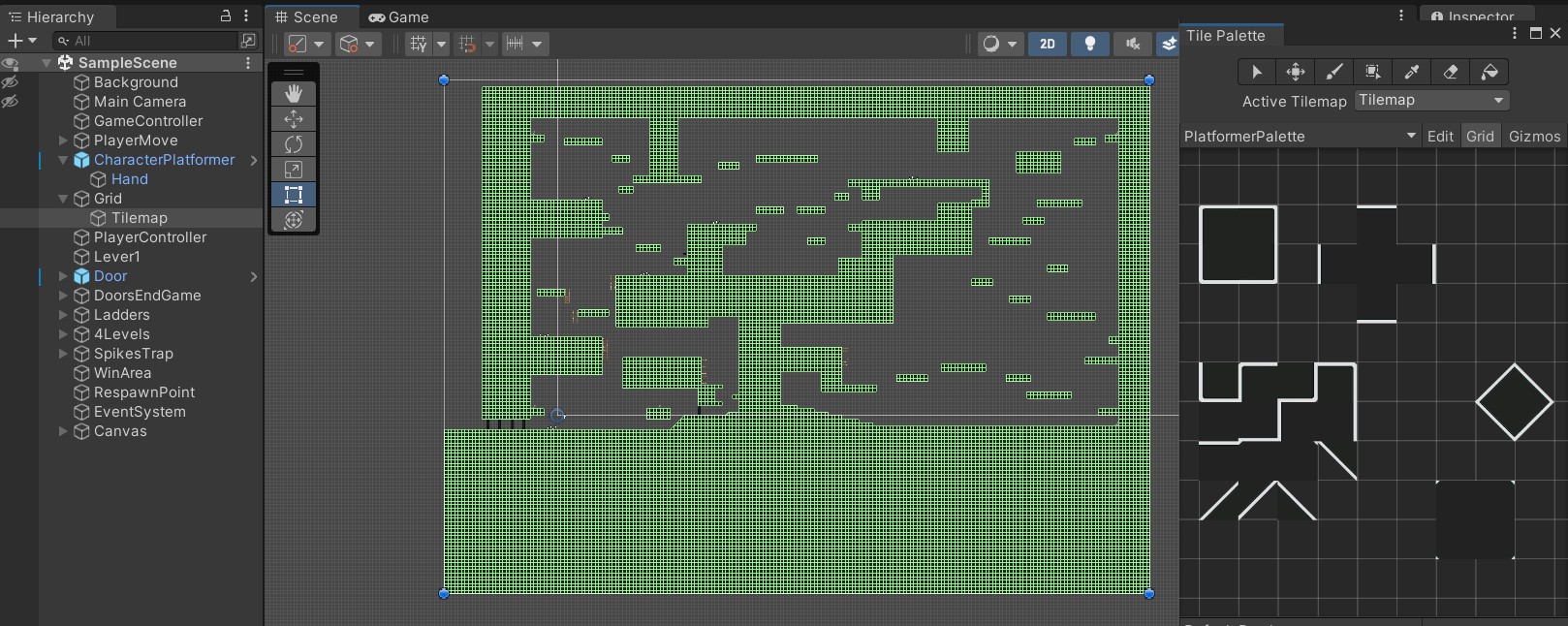
Teleport(last\_ground\_pos);

}

}

Выше приведены 2 основных цикла для работы персонажа игрока.

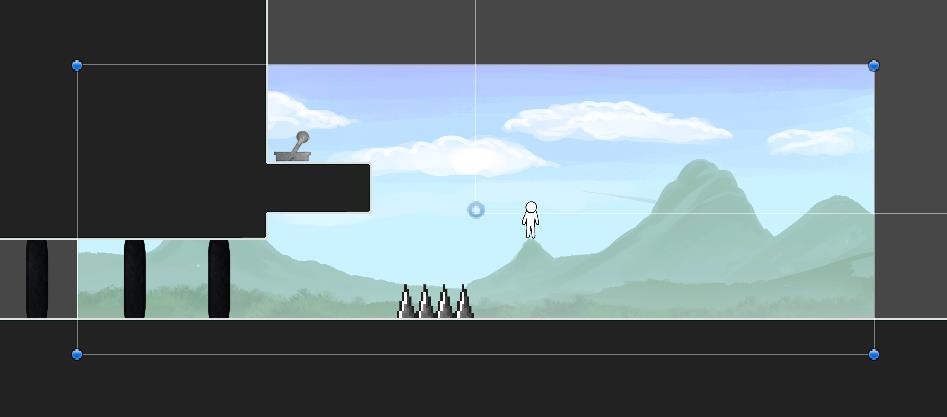
# Главная карта:



Основная карта сделана из тайловой карты, сгенерированной из "GRID" и "TILEPALETTE".

В Tilemap встроен коллайдер, поэтому он может сталкиваться с «Rigidbody2D» игрока, чтобы игрок мог стоять на местности.

# Following Background:



us ng System.Collect ons;

us ng System.Collect ons.Gener c; us ng Un tyEng ne;

publ c class BackgroundFollow : MonoBehav our

{

pr vate Vector3 offset = new Vector3(0f, 0f, 10f); [Ser al zeF eld] pr vate Transform camera;

// Update s called once per frame vo d Update()

{

transform.pos t on = camera.pos t on + offset;

}

}

Фон будет автоматически следовать за камерой игрока

# Levels and Doors:



vo d Update()

{

t mer += T me.deltaT me; f (state != prev\_state)

{

ChangeSpr te(); prev\_state = state;

}

}

pr vate vo d OnDestroy()

{

levers.Remove(th s);

}

vo d OnTr ggerEnter2D(Coll der2D coll)

{

f (coll.gameObject.GetComponent<fllayerCharacter>())

{

f (state == LeverState.d sabled) return;

Act vate();

}

}

Включает в себя: Sprite, Collider2D, ... используется для открытия и закрытия дверей



vo d F xedUpdate()

{

//Get nb sw tch on

nt nb\_sw tch = GetNbSw tches();

//keys

nb\_sw tch += nb\_keys\_ ns de;

//Open door

bool act vated = (nb\_sw tch >= nb\_sw tches\_requ red); Vector3 move\_d r = GetMoveD r();

should\_open = opened\_at\_start o !act vated : act vated; target\_pos = transform.pos t on;

f (should\_open)

{

Vector3 d ff = transform.pos t on - n t alflos;

f (open\_speed >= 0.01f && d ff.magn tude < max\_move)

{

}

}

else

{

target\_pos = n t alflos + move\_d r.normal zed \* max\_move; target\_pos.z = 1f;

Vector3 d ff = transform.pos t on - n t alflos; float dot\_prod = Vector3.Dot(d ff, move\_d r);

f (close\_speed >= 0.01f && dot\_prod > 0.001f && d ff.magn tude > 0.01f)

{

target\_pos = n t alflos; target\_pos.z = 1f;

}

}

}

pr vate vo d Update()

{

Vector3 move\_d r = target\_pos - transform.pos t on; f (move\_d r.magn tude > 0.01f)

{

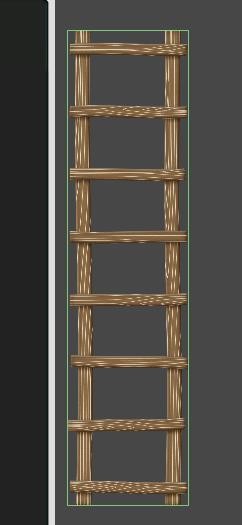
float speed = should\_open o open\_speed : close\_speed;

float move\_d st = Mathf.M n(speed \* T me.deltaT me, move\_d r.magn tude); transform.pos t on += move\_d r.normal zed \* move\_d st;

}

Дверь имеет индекс и может быть разблокирована на соответствующих уровнях.

# Ladders:



f ( sCl mb ng)

{

r g d.grav tyScale = 0f;

r g d.veloc ty = new Vector2(r g d.veloc ty.x, vert cal \* speed); return;

}

else

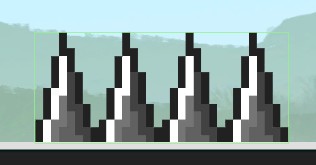
{

r g d.grav tyScale = normalGrav ty;

}

Когда игрок сталкивается с коллайдером Лестницы, игрок сможет подняться, уменьшив вес игрока = 0

# Spikes Trap:



f ( sIns deSp kes)

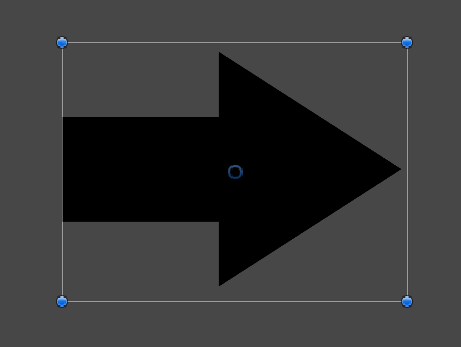
{

TakeDamage(10);

}

Когда игрок входит в коллайдер 2D шипов, игрок получает урон, уменьшенный до полосы HP.

# Mission Arrows:



publ c Transform target;

publ c bool sreachedtarget = false;

// Update s called once per frame vo d Update()

{

f( sreachedtarget)

{

foreach (Transform ch ld n transform)

{

GameObject.Destroy(ch ld.gameObject);

}

Destroy(gameObject); return;

}

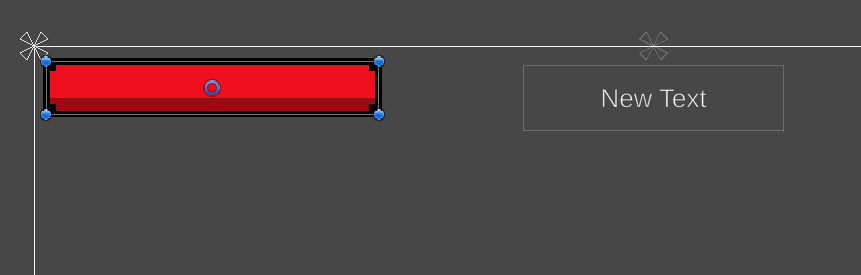
var d r = target.pos t on - transform.pos t on;

var angle = Mathf.Atan2(d r.y, d r.x) \* Mathf.Rad2Deg; transform.rotat on = Ǫuatern on.AngleAx s(angle, Vector3.forward);

}

Стрелка миссии будет вращаться в направлении 4 основных уровней, чтобы помочь игрокам найти их и выиграть игру.

# Canvas:



Canvas включает полосу здоровья и квесты textMeshPro для игроков.

HP Bar — это Slider, и мы можем настроить для него значение.

publ c Sl der sl der\_hp;

publ c vo d TakeDamage(float damage)

{

f (! s\_dead && ! nvulnerable && h t\_t mer > 0f)

{

hp -= damage;

h t\_t mer = -1f;

f (hp <= 0f)

{

sl der\_hp.value = 0; K ll();

}

else

{

sl der\_hp.value = hp; f (onH t != null)

onH t.Invoke();

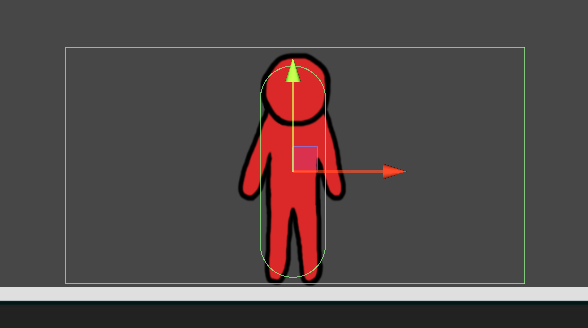
}

}

}

**Новые улучшения для 2-го семестра**

1. **Enemy:**

****

**Если ни один игрок не обнаружен, противник будет свободно перемещаться между двумя предустановленными точками.**

private void Update()

{

if (isPlayerDetected)

{

StopAndAttackPlayer();

}

else

{

MoveToWaypoint();

}

}

**Если игрок обнаружен, противник остановится и атакует игрока, в противном случае противник продолжит движение по 2 обозначенным точкам.**

private void StopAndAttackPlayer()

{

if (player != null)

{

Vector3 moveDirection = player.position - transform.position;

if (moveDirection.x < 0)

{

spriteRenderer.flipX = true;

}

else if (moveDirection.x > 0)

{

spriteRenderer.flipX = false;

}

if (moveDirection.magnitude > attackRange)

{

transform.Translate(moveDirection.normalized \* moveSpeed \* Time.deltaTime);

}

else

{

attackTimer += Time.deltaTime;

if (attackTimer >= attackDelay)

{

AttackPlayer();

attackTimer = 0f;

}

}

}

}

private void AttackPlayer()

{

Debug.Log("Enemy is attacking player.");

playerCharacter.TakeDamage(5);

}

private void MoveToWaypoint()

{

Vector3 targetPosition = waypoints[currentWaypointIndex].position;

moveDirection = targetPosition - transform.position;

transform.Translate(moveDirection.normalized \* moveSpeed \* Time.deltaTime);

if (moveDirection.x < 0)

{

spriteRenderer.flipX = true;

}

else if (moveDirection.x > 0)

{

spriteRenderer.flipX = false;

}

if (Vector3.Distance(transform.position, targetPosition) < 0.1f)

{

currentWaypointIndex = (currentWaypointIndex + 1) % waypoints.Length;

}

}

1. **Coins:**

****

**Монета — это пункт, когда игрок подходит к монете и касается ее, ему добавляется 1 очко, отображаемое на экране.**

private void OnTriggerEnter2D(Collider2D collision)

{

if (collision.CompareTag("Coin"))

{

CollectCoin(collision.gameObject);

}

}

private void CollectCoin(GameObject coin)

{

coin.SetActive(false);

score++;

UpdateScoreUI();

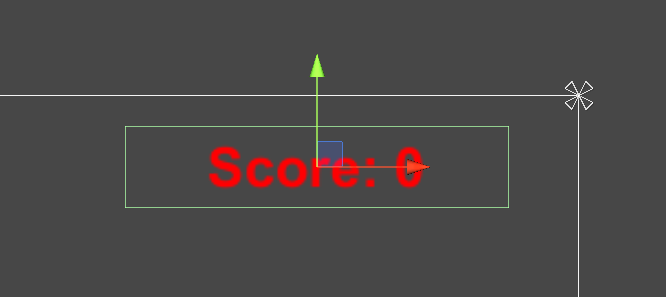
}

private void UpdateScoreUI()

{

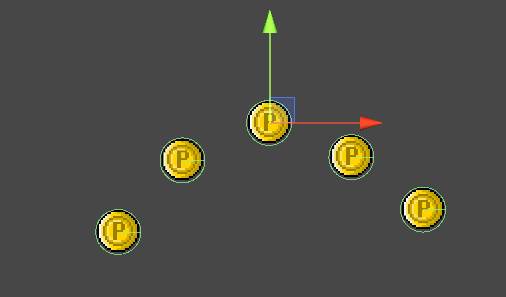
scoreText.text = "Score: " + score.ToString();

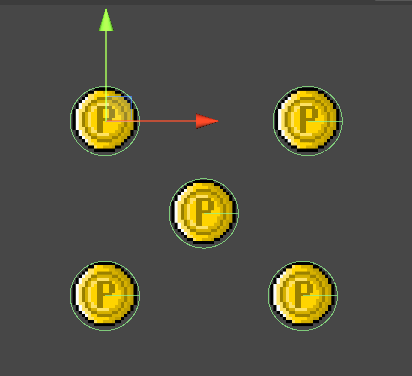
}

****

**Создавайте больше цепочек монет, чтобы сделать игру более интересной.**

****

****

****