государственное автономное учреждение

Калининградской области

профессиональная образовательная организация

«Колледж предпринимательства»

**О Т Ч Е Т**

по производственной практике

ПМ.01 Разработка модулей программного обеспечения для компьютерных систем

Предприятие ООО «МСофт»

Выполнил (а):

студент (ка) 4 курса группы ИСП 19-21

специальность

09.02.07 Информационные системы и

программирование

Родин А. А. й

Фамилия, инициалы

Руководитель Дюжикова А.С.

Дата защиты \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель предприятия Мытник А.А.

М.П.

2023

Содержание

[Введение 3](#_Toc132472201)

[Основная часть 4](#_Toc132472202)

[Вывод 10](#_Toc132472203)

[Список использованных источников 11](#_Toc132472204)

[Приложение 12](#_Toc132472205)

# Введение

В современном мире большую популярность набирает индустрия видео игр, при это самыми популярными направлениями являются многопользовательские, условно бесплатные (с возможностью внутри игровых транзакций) и с элементами коллекционирования виртуальных объектов проекты. Из-за чего сегмент однопользовательских проектов, ориентированных на долгое погружение в историю, изучение механик и бросающих вызов пользователю вымирает. Одними из таких частей стали стратегии – упрощение, попытка ввести внутри игровые транзакции, и концентрация на визуальной части привели к падению продаж, потери целевой аудитории у крупных издателей, и как результат серьёзное падение популярности жанра. Так же проекты данного жанра считаются экономически не выгодными, ввиду больших затрат для производства и малого спроса, так как целевая аудитория не доверяет крупным издателям и внимательно следит за публикуемой информацией о проектах. При этом спрос на стратегии не упал, что послужило хорошей почвой для маленьких команд разработчиков, что выпускают одну игру по шаблонам старых игр или же меняют привычные механики на совсем новые и необычные.

Целью данной работы является создание видео игры в жанре Стратегии, в под жанре RTS, по шаблону старых RTS. А именно:

Создание базовых механик передвижения, управления и стрельбы у игровых пешек.

Создание игрового интерфейса.

Создание механик строительства, ремонта и захвата строений.

Реализация созданных механик и интерфейса в проект.

# Основная часть

Код в файле “Control Unit” отвечает за управление игровой техникой, а именно определить какую технику игрок выбрал и куда нажал на землю, чтобы создать на этом месте точку в которую техника будет передвигаться при помощи navmesh. Так же код сбрасывает статус выбранных у любой техники, если игрок нажал левой клавишей мышки на не технику.

public class ControlUnit : MonoBehaviour

{

public NavMeshAgent agent;

public bool select = false;

public Transform enemy;

// Update is called once per frame

void Update()

{

// Выбор юнита

if (Input.GetMouseButtonDown(0))

{

Ray movePosition = Camera.main.ScreenPointToRay(Input.mousePosition);

if(Physics.Raycast(movePosition, out var hitInfo, Mathf.Infinity, 1<<3))

{

if (hitInfo.collider.gameObject.CompareTag("earth")){}

else if(hitInfo.collider.gameObject.transform.parent.CompareTag("unit"))

{

select = false; hitInfo.rigidbody.gameObject.GetComponentInParent<ControlUnit>().select = true;

}else{

select = false;

}

}

}

if (Input.GetMouseButtonDown(1) && select == true)

{

Ray movePosition = Camera.main.ScreenPointToRay(Input.mousePosition);

if(Physics.Raycast(movePosition, out var hitInfo))

{

agent.SetDestination(hitInfo.point);

}

}

}

}

Код в файлах “artTarget” и “tankTarget” одинаковы за исключением строчки с переменной t, это связано с тем, что в файлах “Turrel” и “TurrelArt” имели разные классы и по которым к ним можно обращаться. Сам код проверяет какие объекты вошли в прозрачную сферу вокруг техники и собирает в список целей все объекты в файле “Storega” параметр команда такого же параметра в технике, к которой файлы “artTarget” и “tankTarget” приклеплены.

tankTarget

public Turrel t;

artTarget

public class artTarget : MonoBehaviour

{

public TurrelArt t;

public List<Transform> AllTar = new List<Transform>();

public GameObject target;

public ControlUnit s;

public int SelectInd = 0;

// Update is called once per frame

void Update()

{

if (AllTar.Count > 0)

{

t.target = AllTar[0];

}

else

{

t.target = null;

}

}

public void OnTriggerEnter(Collider other)

{

if (other.gameObject.GetComponentInParent<storega>() != null && other.gameObject.GetComponentInParent<storega>().team != gameObject.GetComponent<storega>().team)

{

AllTar.Add(other.gameObject.transform.parent.transform);

}

}

public void OnTriggerExit(Collider other)

{

if (other.gameObject.GetComponentInParent<storega>() != null && other.gameObject.GetComponentInParent<storega>().team != gameObject.GetComponent<storega>().team)

{

AllTar.Remove(other.gameObject.transform.parent.transform);

}

}

}

Код файла “Build” создаёт сетку вокруг игрового здания, которая создаётся на основании полученных значений ширены и длины из параметра “Size”. Также два метода: “OnDrawGizmosSelected” и “SetNormal”, первый создаёт кубы, которые видны только разработчику, для отображение занимаемой зданием области, второй перекрашивает здание в исходный цвет, используются в файле “Building”.

public class build : MonoBehaviour

{

public Vector2Int Size = Vector2Int.one;

public Renderer Mainrenderer;

public int costBuild;

public void SetTransparent(bool avaible)

{

if (avaible)

{

Mainrenderer.material.color = Color.green;

}

else

{

Mainrenderer.material.color = Color.red;

}

}

public void SetNormal()

{

Mainrenderer.material.color = Color.white;

}

private void OnDrawGizmosSelected()

{

// Отрисовка сетки здания

for(int x = 0; x < Size.x; x++)

{

for(int y = 0; y < Size.y; y++)

{

Gizmos.color = new Color(0f, 1f, 0f, 0.3f); //Цвет

Gizmos.DrawCube((transform.position) + new Vector3(x, 0, y), new Vector3(1,.1f,1)); //Отрисовка куба

}

}

}

}

Код файлы “Building” (смотрите в приложении) отвечает логику строительства, создание области где можно разместить строение, перемещение здания по полю на одну условную клетку, так как строение перемещается на 1000 единиц измерения в “Unity”, перемещение здания за курсором мышки, осуществляемое при помощи проведение луча от курсора в область куда направлена камера и когда луч касается земли в это место перемещается здание, проверка на возможность строительства, осуществляется проверкой массива “grid”, в котором записаны координаты зданий и сами здания, изменение цвета здания в зависимости от возможности установить его на землю, установка здания на область и запись координат и этого строения в массив.

Код файла “BulletF” отвечает за нанесение урона игровым объектам, осуществляемое обращением к файлу “Storega" и уменьшения параметра “hp” после чего пуля удаляется из игры.

public class bulletF : MonoBehaviour

{

public int damage;

private void OnTriggerEnter(Collider other)

{

if (other != null && other.gameObject.layer == 3)

{

if(other.gameObject.tag != "earth")

{

other.gameObject.GetComponentInParent<storega>().hp -= damage;

}

Destroy(gameObject);

}

}

}

Код файла “factoryScript” (смотрите в приложении) открывает кнопки для создания игровой техники при условии, если игрок выбрал завод. При получении образца техники, создаёт её в точке “pointSpawn”, только если у игрока хватает денег.

Код файла “SpawnBut” привязан к кнопкам создания техники, при их нажатии происходит поиск всех выбранных заводов и передаёт им образец техники для создания.

public class spawnBut : MonoBehaviour

{

public void SpawnBut(GameObject unit)

{

var arr = FindObjectsOfType(typeof(factoryScript));

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

{

if (arr[i].GetComponent<factoryScript>().selected)

{

arr[i].GetComponent<factoryScript>().test = unit;

}

}

}

}

Код файла “playerScript” (смотрите в приложении) перемещает и вращает объект, если игрок нажимает клавиши “ц”, “ф”, “ы”, “в”, “й”, “у”, “к”, “а” на который смотрит камера по игровой сцене и показывает количества игровой валюты.

Файла “Storega" хранит переменные, и если здоровье объекта меньше или равно нулю.

public class storega : MonoBehaviour

{

public int cost;

public int hp;

public string team;

// Update is called once per frame

void Update()

{

if(hp <= 0)

{

Destroy(gameObject);

}

}

}

Код файла “TurrelArt” (смотрите в приложении) отвечает за наведение башни и орудия, создание снаряда и придание ему такой скорости, с которой снаряд долетит до цели. Расчёт происходит по формуле v= √(gx2)/(2(y-tgα\*x) \* cos2α, где v – скорость, g – ускорение свободно падения, у и x это дистанция до цели на осях y и x соответственно, α – это угол под которым снаряд вылетает из ствола.

Код файла “Turrel” (смотрите в приложении) отвечает за наведение башни и орудия на цель осуществляется получением направления на цель, вычислением насколько нужно развернуть башню и орудие, чтобы они смотрели на цель, после чего создаётся снаряд постоянной скоростью. Поворот башни в файле “TurrelArt” осуществляется точно также.

# Заключение

За время практики были реализованы различные игровые механики.

# Список использованных источников

1. Васильев А.Н. “Программирование на C# для начинающих. Основные сведения”
2. Эндрю Стиллмен и Дженнифер Грин “ Эндрю Стиллмен, Дженнифер Грин”
3. Джереми Гибсон Бонд “Unity и C#. Геймдев от идеи до реализации”
4. https://ru.wikipedia.org/wiki/C\_Sharp
5. https://ru.wikipedia.org/wiki/Unreal\_Engine
6. https://ru.wikipedia.org/wiki/Godot
7. https://ru.wikipedia.org/wiki/Игровой\_движок#cite\_note-\_e5e20b91561a71fd-2
8. <https://ru.wikipedia.org/wiki/Unity_(игровой_движок)>
9. <https://en.wikipedia.org/wiki/Navigation_mesh>
10. [https://ru.wikipedia.org/wiki/Компьютерная\_стратегическая\_игра](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D1%81%D1%82%D1%80%D0%B0%D1%82%D0%B5%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0)

# Приложение

Код файлы “Building”

public class building : MonoBehaviour

{

public Vector2Int GridSize = new Vector2Int(400, 400);

private build[,] grid;

private build flyingBuilding;

private Camera mainCamera;

public playerScript player;

private void Awake()

{

grid = new build[GridSize.x, GridSize.y];

mainCamera = Camera.main;

//нахождение игрока

player = FindObjectOfType<playerScript>();

}

//Получение префаба здания

public void StartPlacingBuilding(build buildPref)

{

if(flyingBuilding != null)

{

Destroy(flyingBuilding.gameObject);

}

flyingBuilding = Instantiate(buildPref);

flyingBuilding.gameObject.GetComponent<NavMeshObstacle>().enabled = false;

}

// Update is called once per frame

private void Update()

{

if (flyingBuilding != null)

{

var groundPlane = new Plane(Vector3.up, Vector3.zero);

Ray ray = mainCamera.ScreenPointToRay(Input.mousePosition);

if(groundPlane.Raycast(ray, out float position))

{ //создание сетки(перемещение строения на своюширину)

Vector3 worldPosition = ray.GetPoint(position);

int x = Mathf.RoundToInt(worldPosition.x/100)\*100;

int y = Mathf.RoundToInt(worldPosition.z/100)\*100;

bool available = true;

if(x < 0 || x > GridSize.x - flyingBuilding.Size.x) available = false;

if(y < 0 || y > GridSize.x - flyingBuilding.Size.y) available = false;

if(available && IsPlaceTaker(x, y)) available = false;

flyingBuilding.transform.position = new Vector3(x, 0, y);

flyingBuilding.SetTransparent(available);

if (available && Input.GetMouseButtonDown(0))

{

PlaceFlyingBuilding(x, y);

}

}

}

}

//Проверка свободна ли земля

private bool IsPlaceTaker(int placeX, int placeY)

{

for (int i = 0; i < flyingBuilding.Size.x; i++)

{

for (int y = 0; y < flyingBuilding.Size.y; y++)

{

if(grid[placeX + placeX, placeY] != null)

{

return true;

}

}

}

return false;

}

private void PlaceFlyingBuilding(int placeX, int placeY)

{

if (flyingBuilding.gameObject.GetComponent<build>().costBuild <= player.cash)

{

player.cash -= flyingBuilding.gameObject.GetComponent<build>().costBuild;

for (int i = 0; i < flyingBuilding.Size.x; i++)

{

for (int y = 0; y < flyingBuilding.Size.y; y++)

{

grid[placeX + placeX, placeY] = flyingBuilding;

}

}

flyingBuilding.SetNormal();

flyingBuilding.gameObject.GetComponent<NavMeshObstacle>().enabled = true;

flyingBuilding.gameObject.GetComponent<storega>().team = "player";

flyingBuilding = null;

}

}

}

Код файла “factoryScript”

public class factoryScript : MonoBehaviour

{

public GameObject pointSpawn;

public Canvas canvas;

public GameObject test;

public bool selected = false;

public bool clickInterfase;

public GraphicRaycaster raycaster;

PointerEventData m\_PointerEventData;

EventSystem m\_EventSystem;

private GameObject b;

public playerScript player;

void Start()

{

canvas = FindObjectOfType<Canvas>();

test = null;

//получение графического raycast из конваса

raycaster = canvas.GetComponent<GraphicRaycaster>();

//получение событий систецмы из сцены

m\_EventSystem = GetComponent<EventSystem>();

//нахождение игрока

player = FindObjectOfType<playerScript>();

}

// Update is called once per frame

void Update()

{

if (Input.GetMouseButtonDown(0))

{

// Графический raycast проверяющий обращается ли игрок к интерфейсу

clickInterfase = false;

// устанавливаем новое событие;

m\_PointerEventData = new PointerEventData(m\_EventSystem);

// устанавливаем положение события на положения мышки

m\_PointerEventData.position = Input.mousePosition;

// создаём лист результатов raycast

List<RaycastResult> results = new List<RaycastResult>();

// записываем в лист результаты графического raycast по положению щелчка

raycaster.Raycast(m\_PointerEventData, results);

// обработка результатов

foreach (RaycastResult result in results)

{

clickInterfase = true;

}

Ray movePosition = Camera.main.ScreenPointToRay(Input.mousePosition);

if (Physics.Raycast(movePosition, out var hitInfo, Mathf.Infinity, (1 << 3 ) | (1 << 5)))

{

GameObject q = hitInfo.collider.gameObject;

if (q.CompareTag("factory") || clickInterfase)

{

q.GetComponent<factoryScript>().selected = true;

}

else

{

selected = false;

}

UnityEngine.Debug.Log(q);

for (int i = 2; i < canvas.transform.childCount; i++)

//Смена меню

{

if (q.CompareTag("factory") || clickInterfase)

{

if (canvas.transform.GetChild(i).name != "factoryMenu")

{

canvas.transform.GetChild(i).gameObject.SetActive(false);

}

else

{

canvas.transform.GetChild(i).gameObject.SetActive(true);

}

}

else

{

if (canvas.transform.GetChild(i).name != "factoryMenu")

{

canvas.transform.GetChild(i).gameObject.SetActive(true);

}

else

{

canvas.transform.GetChild(i).gameObject.SetActive(false);

}

}

}

}

}

if (selected && test != null && player.cash >= test.GetComponent<storega>().cost)

{

player.cash -= test.GetComponent<storega>().cost;

b = Instantiate(test, pointSpawn.transform.position, pointSpawn.transform.rotation);

b.GetComponent<storega>().team = gameObject.GetComponent<storega>().team;

test = null;

}

}

}

Код файла “playerScript”

public class playerScript : MonoBehaviour

{

private Rigidbody \_rb;

public float speed = 10f;

public int cash = 1000;

public Text cashIm;

// Start is called before the first frame update

void Start()

{

\_rb = GetComponent<Rigidbody>();

}

// Update is called once per frame

void Update()

{

\_rb.velocity = new Vector3(Input.GetAxis("Horizontal") \* speed, \_rb.velocity.y, Input.GetAxis("Vertical") \* speed);

if (Input.GetKey(KeyCode.Q))

{

Rotaite(0);

}

if (Input.GetKey(KeyCode.E))

{

Rotaite(1);

}

if (Input.GetKey(KeyCode.R))

{

Rotaite(2);

}

if (Input.GetKey(KeyCode.F))

{

Rotaite(3);

}

if (Input.GetKey(KeyCode.Z))

{

}

cashIm.text = cash + "$";

}

public void Rotaite(int lR)

{

if (lR == 0)

{

transform.Rotate(((float)(0.25)), 0, 0);

}

if (lR == 1)

{

transform.Rotate(((float)(-0.25)), 0, 0);

}

if (lR == 2)

{

transform.Rotate(0, ((float)(-0.25)),0, Space.World);

}

if (lR == 3)

{

transform.Rotate(0, ((float)(0.25)),0, Space.World);

}

}

}

Код файла “TurrelArt”

public class TurrelArt : MonoBehaviour

{

public Transform shootEl;

public Transform navBulletPos;

public Transform LookAtObj;

public float dmg = 10;

public float shootSpeed;

public GameObject bullet;

public Transform target;

public bool isShoot;

public GameObject GunTank;

public float SpeedRotation = 90;

public Transform PointRotateGun;

public float AngleInDegrees;

public float g = Physics.gravity.y;

// Start is called before the first frame update

void Start()

{

}

// Update is called once per frame

void Update()

{

//Наведение турели на цель

if(target != null)

{

Vector3 navAtTargetXZ = Vector2d("y"); //Получение направления на цель

Quaternion targetRotation = Quaternion.LookRotation(navAtTargetXZ); //Преобразование направления на цель в поворот на Квартерионы

transform.rotation = Quaternion.RotateTowards(transform.rotation, targetRotation, SpeedRotation \* Time.deltaTime); //Поворот турели на цель каждый кадр

// Балистика

PointRotateGun.localEulerAngles = new Vector3(-AngleInDegrees, 0f, 0f);

//

if (isShoot == false)

{

StartCoroutine(Shoot());

}

}

}

}

public Vector3 Vector2d(string axis)

{

Vector3 nav = target.position - transform.position; //Получение направления на цель

switch (axis)

{

case "x":

nav.x = 0;

break;

case "y":

nav.y = 0;

break;

case "z":

nav.z = 0;

break;

}

return nav;

}

//стрельба

IEnumerator Shoot()

{

isShoot = true;

yield return new WaitForSeconds(shootSpeed);

// Балистика

// изменяя скорость снаряда

Vector3 nav = target.position - shootEl.position;

Vector3 navXZ = new Vector3(nav.x, 0f, nav.z);

float x = navXZ.magnitude;

float y = nav.y;

float AngleInRadians = AngleInDegrees \* Mathf.PI / 180;

float v2 = (g \* x \* x) / (2 \* (y - Mathf.Tan(AngleInRadians) \* x) \* Mathf.Pow(Mathf.Cos(AngleInRadians), 2));

float v = Mathf.Sqrt(Mathf.Abs(v2));

GameObject b = Instantiate(bullet, shootEl.position, shootEl.rotation);

b.GetComponent<Rigidbody>().velocity = shootEl.transform.forward \* v;

//

isShoot = false;

}

}

Код файла “Turrel”

public class Turrel : MonoBehaviour

{

public Transform shootEl;

public Transform navBulletPos;

public Transform LookAtObj;

public float dmg = 10;

public float shootSpeed;

public GameObject bullet;

public Transform target;

public bool isShoot;

public GameObject GunTank;

public float SpeedRotation = 90;

public Transform PointRotateGun;

public float AngleInDegrees;

// Update is called once per frame

void Update()

{

//Наведение турели на цель

if(target != null)

{

Vector3 navAtTargetXZ = Vector2d("y"); //Получение направления на цель

Quaternion targetRotation = Quaternion.LookRotation(navAtTargetXZ, Vector3.up); //Преобразование направления на цель в поворот на Квартерионы

transform.rotation = Quaternion.RotateTowards(transform.rotation, targetRotation, SpeedRotation \* Time.deltaTime); //Поворот турели на цель каждый кадр

// PointRotateGun.rotation = Quaternion.RotateTowards(transform.rotation, targetRotation, SpeedRotation \* Time.deltaTime);

Vector3 navAtTargetYZ = target.position - transform.position;

Quaternion t = Quaternion.LookRotation(navAtTargetYZ);

PointRotateGun.rotation = Quaternion.RotateTowards(PointRotateGun.rotation, t, 10 \* Time.deltaTime);

Ray traicShell = new Ray(shootEl.position, shootEl.forward);

Debug.DrawRay(shootEl.position, shootEl.forward);

RaycastHit hit;

if (Physics.Raycast(traicShell, out hit, Mathf.Infinity, (1 << 3)))

{

if (hit.collider.gameObject.transform.parent.gameObject == target.gameObject)

{

if (isShoot == false)

{

StartCoroutine(Shoot());

}

}

}

}

}

public Vector3 Vector2d(string axis)

{

Vector3 nav = target.position - transform.position; //Получение направления на цель

switch (axis)

{

case "x":

nav.x = 0;

break;

case "y":

nav.y = 0;

break;

case "z":

nav.z = 0;

break;

}

return nav;

}

//стрельба

IEnumerator Shoot()

{

isShoot = true;

yield return new WaitForSeconds(shootSpeed);

Vector3 n = navBulletPos.position - shootEl.position;

GameObject b = GameObject.Instantiate(bullet, shootEl.position, transform.rotation) as GameObject;

b.GetComponent<Rigidbody>().velocity = n \* 1;

isShoot = false;

Debug.Log("qwe");

}

}