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](#_Toc132639929)

[Основная часть 4](#_Toc132639930)

[Заключение 11](#_Toc132639931)

[Список использованных источников 12](#_Toc132639932)

[ПРИЛОЖЕНИЯ 13](#_Toc132639933)

# 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.

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.

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”.

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.

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.

Код файла “playerScript” 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.

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

Производственная практика проходила на предприятии ООО «МСофт». В ходе практики были освоены навыки написания код на языке программирования C#, создания трёхмерных моделей в программе “MagicaVoxel” и оформления документов. Данные навыки пригодились в написании диплома, а именно: создание трёхмерных моделей для диплома; написание программного кода для диплома; написание и оформлении части диплома.

# Список использованных источников

1. <https://www.litres.ru/book/aleksey-nikolaevich/programmirovanie-na-c-dlya-nachinauschih-osnovnye-sve-34338191/>
2. <https://www.litres.ru/book/dzhennifer-grin-2/izuchaem-c-8479459/>
3. <https://www.litres.ru/dzheremi-gibson-bond/unity-i-s-geymdev-ot-idei-do-realizacii-pdf-64085991/>
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)

# ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А

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;

player = FindObjectOfType<playerScript>();

}

public void StartPlacingBuilding(build buildPref)

{

if(flyingBuilding != null)

{

Destroy(flyingBuilding.gameObject);

}

flyingBuilding = Instantiate(buildPref);

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

}

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))

{

Vector3 worldPosition = ray.GetPoint(position);

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

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

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЕ А

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);

}

}

}

}

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;

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЕ А

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;

}

}

}

ПРИЛОЖЕНИЕ Б

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;

raycaster = canvas.GetComponent<GraphicRaycaster>();

m\_EventSystem = GetComponent<EventSystem>();

player = FindObjectOfType<playerScript>();

}

void Update()

{

if (Input.GetMouseButtonDown(0))

{

clickInterfase = false;

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

m\_PointerEventData.position = Input.mousePosition;

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

графического raycast по положению щелчка

raycaster.Raycast(m\_PointerEventData, results);

foreach (RaycastResult result in results)

{

clickInterfase = true;

}

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

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЕ Б

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++)

{

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

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЕ Б

{

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;

}

}

}

ПРИЛОЖЕНИЕ В

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;

void Update()

{

if(target != null)

{

Vector3 navAtTargetXZ = Vector2d("y");

Quaternion targetRotation = Quaternion.LookRotation(navAtTargetXZ); Квартерионы

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

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;

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЕ В

break;

case "y":

nav.y = 0;

break;

case "z":

nav.z = 0;

break;

}

return nav;

}

IEnumerator Shoot()

{

isShoot = true;

yield return new WaitForSeconds(shootSpeed);

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;

}

}

ПРИЛОЖЕНИЕ Г

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;

void Update()

{

if(target != null)

{

Vector3 navAtTargetXZ = Vector2d("y");

Quaternion targetRotation = Quaternion.LookRotation(navAtTargetXZ, Vector3.up); Квартерионы

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

// 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)))

ПРОДОЛЖЕНИЕ ПРИЛОЖЕНИЕ Г

{

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;

switch (axis)

{

case "x":

nav.x = 0;

break;

case "y":

nav.y = 0;

break;

case "z":

nav.z = 0;

break;

}

return nav;

}

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");

}

}