**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |  |
| --- | --- | --- |
| СОГЛАСОВАНО  Старший преподаватель  департамента  программной инженерии  факультета компьютерных наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О.В. Максименкова  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 г. |  | УТВЕРЖДАЮ  Академический руководитель образовательной программы «Программная инженерия»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шилов  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 г. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** | RU.17701729.503200-01 12 01-1 | | **МОБИЛЬНАЯ ИГРА  «ЗМЕЕГУСЕНИЦА»**  **Текст программы**  **ЛИСТ УТВЕРЖДЕНИЯ**  **RU.17701729.** **508830-01 12 01-1** | | |
|  |  | |
| Исполнитель:  студентка группы БПИ165  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Петрова А.В. /  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2017 г. | |
|  | | |
|  | |  |

**2017**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** | RU.17701729.508830-01 12 01-1 | | **МОБИЛЬНАЯ ИГРА «ЗМЕЕГУСЕНИЦА»**  **Текст программы**  **RU.17701729.** **508830-01 12 01-1**  **Листов 45** | | | | |
|  |  | | | |
|  | | | |
|  | | | | |
|  | | |  | |

**2017**

СОДЕРЖАНИЕ

[1. ТЕКСТ ПРОГРАММЫ 3](#_Toc482866885)

[1.1. AudioController 3](#_Toc482866886)

[1.2. Bot 3](#_Toc482866887)

[1.3. ChangeScene 7](#_Toc482866888)

[1.4. ChangeTurnsCount 7](#_Toc482866889)

[1.5. ColorChange 8](#_Toc482866890)

[1.6. ColorsController 9](#_Toc482866891)

[1.7. Controller 10](#_Toc482866892)

[1.8. GameController 11](#_Toc482866893)

[1.9. GameVsComputerController 18](#_Toc482866894)

[1.10. GooglePlayScript 19](#_Toc482866895)

[1.11. InitializeCustomization 21](#_Toc482866896)

[1.12. InitializeGameSettings 22](#_Toc482866897)

[1.13. InitializeSet 23](#_Toc482866898)

[1.14. Link 23](#_Toc482866899)

[1.15. Nod 24](#_Toc482866900)

[1.16. NotificationManager 24](#_Toc482866901)

[1.17. PageSwitcher 25](#_Toc482866902)

[1.18. Panel 26](#_Toc482866903)

[1.19. Rotate 27](#_Toc482866904)

[1.20. Settings 28](#_Toc482866905)

[1.21. SkinsController 30](#_Toc482866906)

[1.22. Switcher 31](#_Toc482866907)

[1.23. Tutorial 31](#_Toc482866908)

[1.24. UnitController 32](#_Toc482866909)

[1.25. UnitControllerWeb 35](#_Toc482866910)

[1.26. WebGameController 38](#_Toc482866911)

[1.27. WebGameManager 40](#_Toc482866912)

[1.28. WebGameUI 44](#_Toc482866913)

[ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ 45](#_Toc482866914)

1. **ТЕКСТ ПРОГРАММЫ**

## AudioController

using UnityEngine;

/// <summary>

/// Скрипт для кнопок управления звуком

/// </summary>

public class AudioController : MonoBehaviour

{

// Update is called once per frame

public void SetAudioState(string tag)

{

if (tag == "music")

FindObjectOfType<Settings>().SetMusic();

if (tag == "sound")

FindObjectOfType<Settings>().SetSound();

}

}

## Bot

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

/// <summary>

/// Бот для игры против компьютера

/// </summary>

public class Bot : MonoBehaviour

{

public GameVsComputerController GameControl;

public int PrevX;

public int PrevY;

void Start()

{

GameControl = FindObjectOfType<GameVsComputerController>();

PrevX = 0;

PrevY = 0;

}

public IEnumerator MakeMoves()

{

yield return new WaitForSeconds(2);

while (GameControl.gameField.Caterpillar.Turns > 0 & GameControl.gameField.AvailableMoves())

{

print("ходов: " + GameControl.gameField.Caterpillar.Turns);

int x = GameControl.gameField.Caterpillar.ActiveX;

int y = GameControl.gameField.Caterpillar.ActiveY;

List<GameObject> possmoves = new List<GameObject>();

try

{

if (GameControl.gameField.unit[x + 1, y].gameObject.tag == "Empty" || GameControl.gameField.unit[x + 1, y].gameObject.tag == "Food")

if (GameControl.gameField.Caterpillar.UnitIsOkToAdd(x + 1, y) && GameControl.gameField.TouchOnlyOneBodyElement(x + 1, y))

{

possmoves.Add(GameControl.gameField.unit[x + 1, y]);

//print("add");

}

}

catch { }

try

{

if (GameControl.gameField.unit[x - 1, y].gameObject.tag == "Empty" || GameControl.gameField.unit[x - 1, y].gameObject.tag == "Food")

if (GameControl.gameField.Caterpillar.UnitIsOkToAdd(x - 1, y) && GameControl.gameField.TouchOnlyOneBodyElement(x - 1, y))

{

possmoves.Add(GameControl.gameField.unit[x - 1, y]);

//print("add");

}

}

catch { }

try

{

if (GameControl.gameField.unit[x, y + 1].gameObject.tag == "Empty" || GameControl.gameField.unit[x, y + 1].gameObject.tag == "Food")

if (GameControl.gameField.Caterpillar.UnitIsOkToAdd(x, y + 1) && GameControl.gameField.TouchOnlyOneBodyElement(x, y + 1))

{

possmoves.Add(GameControl.gameField.unit[x, y + 1]);

//print("add");

}

}

catch { }

try

{

if (GameControl.gameField.unit[x, y - 1].gameObject.tag == "Empty" || GameControl.gameField.unit[x, y - 1].gameObject.tag == "Food")

if (GameControl.gameField.Caterpillar.UnitIsOkToAdd(x, y - 1) && GameControl.gameField.TouchOnlyOneBodyElement(x, y - 1))

{

possmoves.Add(GameControl.gameField.unit[x, y - 1]);

//print("add");

}

}

catch { }

List<GameObject> completedmoves = new List<GameObject>();

foreach(GameObject possmove in possmoves)

{

int possmoveX = possmove.GetComponent<UnitController>().x;

int possmoveY = possmove.GetComponent<UnitController>().y;

if (AllowsMakeAllMoves(GameControl.gameField.Caterpillar.Turns - 1, possmoveX, possmoveY, GameControl.gameField.Caterpillar.ActiveX, GameControl.gameField.Caterpillar.ActiveY, PrevX, PrevY))

{

completedmoves.Add(possmove);

}

}

print("possmoves.Count: " + possmoves.Count);

print("completedmoves.Count: " + completedmoves.Count);

if (completedmoves.Count != 0)

{

completedmoves[Random.Range(0, completedmoves.Count)].GetComponent<UnitController>().PlaceElementForBot();

}

else

{

possmoves[Random.Range(0, possmoves.Count)].GetComponent<UnitController>().PlaceElementForBot();

}

yield return new WaitForSeconds(0.5F);

}

}

bool AllowsMakeAllMoves(int moves, int activeX, int activeY, int prev1X, int prev1Y, int prev2X, int prev2Y)

{

bool thisposition = false;

//нижний элемент

if (moves == 0)

return true;

try

{

if (moves == 1 && GameControl.gameField.unit[activeX + 1, activeY].gameObject.tag == "Empty")

{

if (UnitIsOkToAdd(activeX + 1, activeY, prev1X, prev1Y, prev2X, prev2Y) && TouchNoneBodyElement(activeX + 1, activeY))

return true;

}

else

{

if (GameControl.gameField.unit[activeX + 1, activeY].gameObject.tag == "Empty")

if (UnitIsOkToAdd(activeX + 1, activeY, prev1X, prev1Y, prev2X, prev2Y) && TouchNoneBodyElement(activeX + 1, activeY))

thisposition |= AllowsMakeAllMoves(moves - 1, activeX + 1, activeY, activeX, activeY, prev1X, prev1Y);

if (GameControl.gameField.unit[activeX + 1, activeY].gameObject.tag == "Food")

if (UnitIsOkToAdd(activeX + 1, activeY, prev1X, prev1Y, prev2X, prev2Y) && TouchNoneBodyElement(activeX + 1, activeY))

thisposition |= AllowsMakeAllMoves(moves - 1 + 2, activeX + 1, activeY, activeX, activeY, prev1X, prev1Y);

}

}

catch { }

//верхний элемент

try

{

if (moves == 1 && GameControl.gameField.unit[activeX - 1, activeY].gameObject.tag == "Empty")

{

if (UnitIsOkToAdd(activeX - 1, activeY, prev1X, prev1Y, prev2X, prev2Y) && TouchNoneBodyElement(activeX - 1, activeY))

return true;

}

else

{

if (GameControl.gameField.unit[activeX - 1, activeY].gameObject.tag == "Empty")

if (UnitIsOkToAdd(activeX - 1, activeY, prev1X, prev1Y, prev2X, prev2Y) && TouchNoneBodyElement(activeX - 1, activeY))

thisposition |= AllowsMakeAllMoves(moves - 1, activeX - 1, activeY, activeX, activeY, prev1X, prev1Y);

if (GameControl.gameField.unit[activeX - 1, activeY].gameObject.tag == "Food")

if (UnitIsOkToAdd(activeX - 1, activeY, prev1X, prev1Y, prev2X, prev2Y) && TouchNoneBodyElement(activeX - 1, activeY))

thisposition |= AllowsMakeAllMoves(moves - 1 + 2, activeX - 1, activeY, activeX, activeY, prev1X, prev1Y);

}

}

catch { }

//правый элемент

try

{

if (moves == 1 && GameControl.gameField.unit[activeX, activeY + 1].gameObject.tag == "Empty")

{

if (UnitIsOkToAdd(activeX, activeY + 1, prev1X, prev1Y, prev2X, prev2Y) && TouchNoneBodyElement(activeX, activeY + 1))

return true;

}

else

{

if (GameControl.gameField.unit[activeX, activeY + 1].gameObject.tag == "Empty")

if (UnitIsOkToAdd(activeX, activeY + 1, prev1X, prev1Y, prev2X, prev2Y) && TouchNoneBodyElement(activeX, activeY + 1))

thisposition |= AllowsMakeAllMoves(moves - 1, activeX, activeY + 1, activeX, activeY, prev1X, prev1Y);

if (GameControl.gameField.unit[activeX, activeY + 1].gameObject.tag == "Food")

if (UnitIsOkToAdd(activeX, activeY + 1, prev1X, prev1Y, prev2X, prev2Y) && TouchNoneBodyElement(activeX, activeY + 1))

thisposition |= AllowsMakeAllMoves(moves - 1 + 2, activeX, activeY + 1, activeX, activeY, prev1X, prev1Y);

}

}

catch { }

//левый элемент

try

{

if (moves == 1 && GameControl.gameField.unit[activeX, activeY - 1].gameObject.tag == "Empty")

{

return (UnitIsOkToAdd(activeX, activeY - 1, prev1X, prev1Y, prev2X, prev2Y) && TouchNoneBodyElement(activeX, activeY - 1));

}

else

{

if (GameControl.gameField.unit[activeX, activeY - 1].gameObject.tag == "Empty")

if (UnitIsOkToAdd(activeX, activeY - 1, prev1X, prev1Y, prev2X, prev2Y) && TouchNoneBodyElement(activeX, activeY - 1))

thisposition |= AllowsMakeAllMoves(moves - 1, activeX, activeY - 1, activeX, activeY, prev1X, prev1Y);

if (GameControl.gameField.unit[activeX, activeY - 1].gameObject.tag == "Food")

if (UnitIsOkToAdd(activeX, activeY - 1, prev1X, prev1Y, prev2X, prev2Y) && TouchNoneBodyElement(activeX, activeY - 1))

thisposition |= AllowsMakeAllMoves(moves - 1 + 2, activeX, activeY - 1, activeX, activeY, prev1X, prev1Y);

}

}

catch { }

return thisposition;

}

public bool UnitIsOkToAdd(int x, int y, int prev1X, int prev1Y, int prev2X, int prev2Y)

{

if (x == prev1X && y == prev1Y)

return false;

if (x == prev1X + 1 || x == prev1X - 1)

if (y == prev1Y)

return false;

if (y == prev1Y + 1 || y == prev1Y - 1)

if (x == prev1X)

return false;

if (x == prev2X + 1 || x == prev2X - 1)

if (y == prev2Y)

return false;

if (y == prev2Y + 1 || y == prev2Y - 1)

if (x == prev2X)

return false;

return true;

}

bool TouchNoneBodyElement(int x, int y)

{

try

{

if (GameControl.gameField.unit[x + 1, y].gameObject.tag == "BodyElement" || GameControl.gameField.unit[x + 1, y].gameObject.tag == "Head")

return false;

}

catch { }

try

{

if (GameControl.gameField.unit[x - 1, y].gameObject.tag == "BodyElement" || GameControl.gameField.unit[x - 1, y].gameObject.tag == "Head")

return false;

}

catch { }

try

{

if (GameControl.gameField.unit[x, y + 1].gameObject.tag == "BodyElement" || GameControl.gameField.unit[x, y + 1].gameObject.tag == "Head")

return false;

}

catch { }

try

{

if (GameControl.gameField.unit[x, y - 1].gameObject.tag == "BodyElement" || GameControl.gameField.unit[x, y - 1].gameObject.tag == "Head")

return false;

}

catch { }

return true;

}

}

## ChangeScene

using UnityEngine;

using UnityEngine.SceneManagement;

public class ChangeScene : StateMachineBehaviour {

public string Scene;

override void OnStateEnter(Animator animator, AnimatorStateInfo stateInfo, int layerIndex) {

SceneManager.LoadScene(Scene);

}

}

## ChangeTurnsCount

using UnityEngine;

using UnityEngine.UI;

public class ChangeTurnsCount : StateMachineBehaviour {

public int mode;

GameObject Avatar;

bool find = true;

// OnStateExit is called when a transition ends and the state machine finishes evaluating this state

// Смена хода

override void OnStateExit(Animator animator, AnimatorStateInfo stateInfo, int layerIndex) {

if (mode == 1)

{

FindObjectOfType<GameController>().gameField.Caterpillar.Turns = 3;

FindObjectOfType<GameController>().gameField.ChangingPlayer = false;

GameObject.Find("Avatar").GetComponent<Image>().color = FindObjectOfType<GameController>().gameField.Caterpillar.colorToSet;

}

if (mode == 2)

{

if (find)

{

Avatar = GameObject.Find("Avatar");

find = false;

}

FindObjectOfType<GameVsComputerController>().gameField.Caterpillar.Turns = 3;

FindObjectOfType<GameVsComputerController>().gameField.ChangingPlayer = false;

Avatar.SetActive(!Avatar.activeSelf);

}

if (mode == 3)

{

FindObjectOfType<WebGameController>().gameField.Caterpillar.Turns = 3;

FindObjectOfType<WebGameController>().gameField.ChangingPlayer = false;

GameObject.Find("Avatar").GetComponent<Image>().color = FindObjectOfType<WebGameController>().gameField.Caterpillar.colorToSet;

}

}

}

## ColorChange

using UnityEngine;

using UnityEngine.UI;

/// <summary>

/// Управление цветом фона во время матча

/// </summary>

public class ColorChange : MonoBehaviour {

float duration = 1f;

float t = 0;

bool start;

Color color1;

Color color2;

// Use this for initialization

void Start () {

start = false;

}

// Update is called once per frame

void Update () {

if (start)

ColorChanger();

}

public void SetColor(Color color)

{

FindObjectOfType<Camera>().backgroundColor = color;

GetComponent<Image>().color = color;

}

/// <summary>

/// запускает смену цвета

/// </summary>

/// <param name="newcolor"></param>

public void StartColorChange(Color newcolor)

{

color1 = GetComponent<Image>().color;

color2 = newcolor;

t = 0;

start = true;

}

/// <summary>

/// смена цвета

/// </summary>

void ColorChanger()

{

Color color = Color.Lerp(color1, color2, t);

FindObjectOfType<Camera>().backgroundColor = color;

GetComponent<Image>().color = color;

if (t < 1)

{

t += Time.deltaTime / duration;

}

else

{

}

}

}

## ColorsController

using UnityEngine;

/// <summary>

/// Скрипт для кнопок переключения цветовой темы

/// </summary>

public class ColorsController : MonoBehaviour {

public string colorThemeName;

public Color colorToSet1;

public Color colorToSet2;

public GameObject Chosen;

// Update is called once per frame

public void SetColor () {

Chosen.transform.position = gameObject.transform.position;

FindObjectOfType<Settings>().SetColor(colorToSet1, colorToSet2, colorThemeName);

}

}

## Controller

using GooglePlayGames;

using UnityEngine;

using UnityEngine.UI;

/// <summary>

/// Скрипт ячейки поля во время локального мультиплеера

/// </summary>

public class Controller : MonoBehaviour {

public int x;

public int y;

public void AttemptToPlaceElement()

{

if (!FindObjectOfType<GameController>().gameField.ChangingPlayer)

{

if (gameObject.tag == "Empty" || gameObject.tag == "Food")

{

if (FindObjectOfType<GameController>().gameField.Caterpillar.UnitIsOkToAdd(x, y) && FindObjectOfType<GameController>().gameField.TouchOnlyOneBodyElement(x, y))

{

print("set");

FindObjectOfType<GameController>().gameField.PlaySound();

if (gameObject.tag == "Food")

FindObjectOfType<GameController>().gameField.Caterpillar.Turns += 2;

gameObject.tag = "BodyElement";

transform.GetChild(3).gameObject.SetActive(true);

FindObjectOfType<GameController>().gameField.Caterpillar.Turns--;

transform.GetChild(0).GetComponent<SpriteRenderer>().color = FindObjectOfType<GameController>().gameField.Caterpillar.colorToSet;

transform.GetChild(4).gameObject.SetActive(false);

FindObjectOfType<GameController>().gameField.Caterpillar.SetActiveUnit(x, y);

if (FindObjectOfType<GameController>().gameField.Caterpillar.Turns == 0)

{

FindObjectOfType<GameController>().gameField.ChangingPlayer = true;

GameObject.FindGameObjectWithTag("Indicator").GetComponent<Animator>().SetTrigger("Change");

if (FindObjectOfType<GameController>().gameField.Caterpillar.colorToSet == FindObjectOfType<GameController>().gameField.Caterpillar.Player1)

FindObjectOfType<GameController>().gameField.Caterpillar.colorToSet = FindObjectOfType<GameController>().gameField.Caterpillar.Player2;

else FindObjectOfType<GameController>().gameField.Caterpillar.colorToSet = FindObjectOfType<GameController>().gameField.Caterpillar.Player1;

FindObjectOfType<ColorChange>().StartColorChange(FindObjectOfType<GameController>().gameField.Caterpillar.colorToSet);

if (FindObjectOfType<GameController>().gameField.PlayerOnMove == 1)

FindObjectOfType<GameController>().gameField.PlayerOnMove = 2;

else FindObjectOfType<GameController>().gameField.PlayerOnMove = 1;

}

if (!FindObjectOfType<GameController>().gameField.AvailableMoves())

{

PlayGamesPlatform.Instance.IncrementAchievement("CgkI35rhhvsWEAIQAg", 1,

(bool success) => {

// handle success or failure

});

if (FindObjectOfType<GameController>().gameField.PlayerOnMove == 1)

{

GameObject.Find("Winner").GetComponent<Text>().text = "Player 2 won";

GameObject.Find("WinColor").GetComponent<Image>().color = FindObjectOfType<Settings>().GameSettings.ChosenColorPlayer2.GetColor();

}

else

{

GameObject.Find("Winner").GetComponent<Text>().text = "Player 1 won";

GameObject.Find("WinColor").GetComponent<Image>().color = FindObjectOfType<Settings>().GameSettings.ChosenColorPlayer1.GetColor();

}

GameObject.Find("Fade").GetComponent<Animator>().SetTrigger("Cover");

GameObject.Find("Game Over").GetComponent<Animator>().SetTrigger("Appear");

}

return;

}

FindObjectOfType<GameController>().gameField.PlayFail();

}

}

}

}

## GameController

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.EventSystems;

public delegate void ActiveUnitChangedHandler(int currentX, int currentY, int newX, int newY);

public delegate void TurnsCountChangedHandler();

/// <summary>

/// Представляет голову гусеницы

/// </summary>

public class CaterpillarHead

{

public GameObject head;

int x;

int y;

public int X { get { return x; } }

public int Y { get { return y; } }

public CaterpillarHead(ref GameObject[,] arr, int min, int max, string skin)

{

int xt, yt;

for(;;)

{

xt = Random.Range(min, max);

yt = Random.Range(0, 10);

if (arr[xt, yt].tag == "Empty")

break;

}

x = xt;

y = yt;

head = arr[x, y];

arr[x, y].tag = "Head";

arr[x, y].transform.GetChild(2).GetComponent<SpriteRenderer>().sprite = Resources.Load<Sprite>("Game/" + skin + "\_heads");

arr[x,y].transform.GetChild(2).gameObject.SetActive(true);

}

public CaterpillarHead(ref GameObject[,] arr, int xp, int yp)

{

x = xp;

y = yp;

head = arr[x, y];

arr[x, y].tag = "Head";

arr[x, y].transform.GetChild(2).GetComponent<SpriteRenderer>().sprite = Resources.Load<Sprite>("Game/" + "default\_caterpillar\_heads");

arr[x, y].transform.GetChild(2).gameObject.SetActive(true);

}

}

/// <summary>

/// Представляет гусеницу на игровом поле

/// </summary>

public class Caterpillar

{

int turnsCount;

public int Turns

{

get

{

return turnsCount;

}

set

{

turnsCount = value;

TurnsCountChanged();

}

}

public Color Player1;

public Color Player2;

public Color colorToSet;

public CaterpillarHead Head;

public int ActiveX;

public int ActiveY;

public event ActiveUnitChangedHandler ActiveUnitChanged;

public event TurnsCountChangedHandler TurnsCountChanged;

public Caterpillar(CaterpillarHead head, Color p1, Color p2)

{

Head = head;

ActiveX = Head.X;

ActiveY = Head.Y;

Head.head.transform.GetChild(0).GetComponent<SpriteRenderer>().color = new Color(156F / 255F, 249F / 255F, 183F / 255F);

turnsCount = 3;

Player1 = p1;

Player2 = p2;

colorToSet = Player1;

}

public bool UnitIsOkToAdd(int x, int y)

{

if (x == ActiveX + 1 || x == ActiveX - 1)

if (y == ActiveY)

return true;

if (y == ActiveY + 1 || y == ActiveY - 1)

if (x == ActiveX)

return true;

return false;

}

public void SetActiveUnit(int x, int y)

{

ActiveUnitChanged(ActiveX, ActiveY, x, y);

ActiveX = x;

ActiveY = y;

}

}

/// <summary>

/// Представляет еду на игровом поле

/// </summary>

public class FoodClass

{

public GameObject Food;

int x;

int y;

public int X { get { return x; } }

public int Y { get { return y; } }

public FoodClass(ref GameObject[,] arr)

{

int xt, yt;

for (;;)

{

xt = Random.Range(1, 14);

yt = Random.Range(1, 9);

if (arr[xt, yt].tag == "Empty" & arr[xt + 1, yt].tag == "Empty" & arr[xt - 1, yt].tag == "Empty" & arr[xt, yt + 1].tag == "Empty" & arr[xt, yt - 1].tag == "Empty")

break;

}

x = xt;

y = yt;

Food = arr[x, y];

arr[x, y].tag = "Food";

arr[x, y].transform.GetChild(4).gameObject.SetActive(true);

}

public FoodClass(ref GameObject[,] arr, int xp, int yp)

{

x = xp;

y = yp;

Food = arr[x, y];

arr[x, y].tag = "Food";

arr[x, y].transform.GetChild(4).gameObject.SetActive(true);

}

}

/// <summary>

/// Представляет игровое поле

/// </summary>

public class GameFieldPresentation

{

public GameObject[,] unit;

public Caterpillar Caterpillar;

public FoodClass[] Food;

public Settings Set;

public int PlayerOnMove;

public bool ChangingPlayer;

public void PlaySound()

{

Set.sound.Play();

}

public void PlayFail()

{

Set.fail.Play();

}

/// <summary>

/// Конструктор для локального мультиплеера

/// </summary>

/// <param name="arr"></param>

/// <param name="Set"></param>

/// <param name="c1"></param>

/// <param name="c2"></param>

/// <param name="mode"> = номеру сцены </param>

public GameFieldPresentation(GameObject[,] arr, GameObject Set, Color c1, Color c2, int mode)

{

this.Set = Set.GetComponent<Settings>();

unit = arr;

if (mode == 1)

for (int i = 0; i < 15; i++)

{

for (int j = 0; j < 10; j++)

{

unit[i, j].name = "(" + i.ToString() + "," + j.ToString() + ")";

unit[i, j].GetComponent<Controller>().x = i;

unit[i, j].GetComponent<Controller>().y = j;

}

}

if (mode == 2)

for (int i = 0; i < 15; i++)

{

for (int j = 0; j < 10; j++)

{

unit[i, j].name = "(" + i.ToString() + "," + j.ToString() + ")";

unit[i, j].GetComponent<UnitController>().x = i;

unit[i, j].GetComponent<UnitController>().y = j;

}

}

Caterpillar = new Caterpillar(new CaterpillarHead(ref unit, 0, 14, Set.GetComponent<Settings>().GameSettings.ChosenCaterpillar), c1, c2);

Caterpillar.ActiveUnitChanged += ActiveUnitChange;

PlayerOnMove = 1;

ChangingPlayer = false;

Food = new FoodClass[7];

for (int i = 0; i < 7; i++)

Food[i] = new FoodClass(ref unit);

}

/// <summary>

/// Конктруктор для игры по сети

/// </summary>

/// <param name="arr"></param>

/// <param name="Set"></param>

/// <param name="c1"></param>

/// <param name="c2"></param>

public GameFieldPresentation(GameObject[,] arr, GameObject Set, Color c1, Color c2)

{

this.Set = Set.GetComponent<Settings>();

unit = arr;

for (int i = 0; i < 15; i++)

{

for (int j = 0; j < 10; j++)

{

unit[i, j].name = "(" + i.ToString() + "," + j.ToString() + ")";

unit[i, j].GetComponent<UnitControllerWeb>().x = i;

unit[i, j].GetComponent<UnitControllerWeb>().y = j;

}

}

Caterpillar = new Caterpillar(new CaterpillarHead(ref unit, 0, 14, "default\_caterpillar"), c1, c2);

Caterpillar.ActiveUnitChanged += ActiveUnitChange;

PlayerOnMove = 1;

ChangingPlayer = false;

Food = new FoodClass[7];

for (int i = 0; i < 7; i++)

Food[i] = new FoodClass(ref unit);

}

/// <summary>

/// Конструктор, воссоздающий поле по данной конфигурации

/// </summary>

/// <param name="arr"></param>

/// <param name="Set"></param>

/// <param name="c1"></param>

/// <param name="c2"></param>

/// <param name="config"></param>

public GameFieldPresentation(GameObject[,] arr, GameObject Set, Color c1, Color c2, byte[] config)

{

this.Set = Set.GetComponent<Settings>();

unit = arr;

for (int i = 0; i < 15; i++)

{

for (int j = 0; j < 10; j++)

{

unit[i, j].name = "(" + i.ToString() + "," + j.ToString() + ")";

unit[i, j].GetComponent<UnitControllerWeb>().x = i;

unit[i, j].GetComponent<UnitControllerWeb>().y = j;

}

}

Caterpillar = new Caterpillar(new CaterpillarHead(ref unit, config[1], config[2]), c1, c2);

Caterpillar.ActiveUnitChanged += ActiveUnitChange;

PlayerOnMove = 1;

ChangingPlayer = false;

Food = new FoodClass[7];

Food[0] = new FoodClass(ref unit, config[3], config[4]);

Food[1] = new FoodClass(ref unit, config[5], config[6]);

Food[2] = new FoodClass(ref unit, config[7], config[8]);

Food[3] = new FoodClass(ref unit, config[9], config[10]);

Food[4] = new FoodClass(ref unit, config[11], config[12]);

Food[5] = new FoodClass(ref unit, config[13], config[14]);

Food[6] = new FoodClass(ref unit, config[15], config[16]);

}

/// <summary>

/// Конфигурация созданного игрового поля

/// </summary>

/// <returns>байтовый массив для отправки сообщения</returns>

public byte[] GetFieldConfig()

{

byte[] config = new byte[17]; //2\*(1 + 7) + 1

config[0] = (byte)'I';

config[1] = (byte) Caterpillar.Head.X;

config[2] = (byte)Caterpillar.Head.Y;

config[3] = (byte)Food[0].X;

config[4] = (byte)Food[0].Y;

config[5] = (byte)Food[1].X;

config[6] = (byte)Food[1].Y;

config[7] = (byte)Food[2].X;

config[8] = (byte)Food[2].Y;

config[9] = (byte)Food[3].X;

config[10] = (byte)Food[3].Y;

config[11] = (byte)Food[4].X;

config[12] = (byte)Food[4].Y;

config[13] = (byte)Food[5].X;

config[14] = (byte)Food[5].Y;

config[15] = (byte)Food[6].X;

config[16] = (byte)Food[6].Y;

return config;

}

public void ActiveUnitChange(int currentX, int currentY, int newX, int newY)

{

unit[currentX, currentY].transform.GetChild(1).gameObject.SetActive(false);

unit[newX, newY].transform.GetChild(1).gameObject.SetActive(true);

}

public bool AvailableMoves()

{

int x = Caterpillar.ActiveX;

int y = Caterpillar.ActiveY;

try

{

if (unit[x + 1, y].tag == "Empty" || unit[x + 1, y].tag == "Food")

if (Caterpillar.UnitIsOkToAdd(x + 1, y) && TouchOnlyOneBodyElement(x + 1, y))

return true;

}

catch { }

try

{

if (unit[x - 1, y].tag == "Empty" || unit[x - 1, y].tag == "Food")

if (Caterpillar.UnitIsOkToAdd(x - 1, y) && TouchOnlyOneBodyElement(x - 1, y))

return true;

}

catch { }

try

{

if (unit[x, y + 1].tag == "Empty" || unit[x, y + 1].tag == "Food")

if (Caterpillar.UnitIsOkToAdd(x, y + 1) && TouchOnlyOneBodyElement(x, y + 1))

return true;

}

catch { }

try

{

if (unit[x, y - 1].tag == "Empty" || unit[x, y - 1].tag == "Food")

if (Caterpillar.UnitIsOkToAdd(x, y - 1) && TouchOnlyOneBodyElement(x, y - 1))

return true;

}

catch { }

return false;

}

public bool TouchOnlyOneBodyElement(int x, int y)

{

int t = 0;

try

{

if (unit[x + 1, y].tag == "BodyElement" || unit[x + 1, y].tag == "Head")

t++;

}

catch { }

try

{

if (unit[x - 1, y].tag == "BodyElement" || unit[x - 1, y].tag == "Head")

t++;

}

catch { }

try

{

if (unit[x, y + 1].tag == "BodyElement" || unit[x, y + 1].tag == "Head")

t++;

}

catch { }

try

{

if (unit[x, y - 1].tag == "BodyElement" || unit[x, y - 1].tag == "Head")

t++;

}

catch { }

if (t == 1)

return true;

return false;

}

}

/// <summary>

/// Скрипт для локального мультиплеера

/// </summary>

public class GameController : MonoBehaviour {

GameObject[,] unit;

public GameFieldPresentation gameField;

public GameObject prefab;

public GameObject TurnsCount;

// Use this for initialization

void Start ()

{

unit = new GameObject[15, 10];

for(int i = 0; i < 15; i++)

{

for(int j = 0; j < 10; j++)

{

unit[i, j] = Instantiate(prefab, new Vector3(-2.25F + 0.5F\*j, 3F - 0.5F \*i,0), Quaternion.identity);

}

}

gameField = new GameFieldPresentation(unit, FindObjectOfType<Settings>().gameObject, FindObjectOfType<Settings>().GameSettings.ChosenColorPlayer1.GetColor(), FindObjectOfType<Settings>().GameSettings.ChosenColorPlayer2.GetColor(), 1);

gameField.Caterpillar.TurnsCountChanged += UpdateTurnsCount;

TurnsCount.GetComponent<Text>().text = gameField.Caterpillar.Turns.ToString();

GameObject.Find("Avatar").GetComponent<Image>().color = gameField.Caterpillar.colorToSet;

GameObject.FindGameObjectWithTag("Indicator").GetComponent<Animator>().SetTrigger("Show");

FindObjectOfType<ColorChange>().StartColorChange(gameField.Caterpillar.colorToSet);

}

void UpdateTurnsCount()

{

TurnsCount.GetComponent<Text>().text = gameField.Caterpillar.Turns.ToString();

}

}

## GameVsComputerController

using UnityEngine;

using UnityEngine.UI;

/// <summary>

/// Скрипт для игры против компьютера

/// </summary>

public class GameVsComputerController : MonoBehaviour {

GameObject[,] unit;

public GameFieldPresentation gameField;

public GameObject prefab;

public GameObject TurnsCount;

public int Turns;

public int Apples;

public int PlayerOnMove

{

get

{

return gameField.PlayerOnMove;

}

set

{

gameField.PlayerOnMove = value;

if (value == 2)

{

//print(value);

StartCoroutine(FindObjectOfType<Bot>().MakeMoves());

}

}

}

void Start () {

Apples = 0;

unit = new GameObject[15, 10];

for (int i = 0; i < 15; i++)

{

for (int j = 0; j < 10; j++)

{

unit[i, j] = Instantiate(prefab, new Vector3(-2.25F + 0.5F \* j, 3F - 0.5F \* i, 0), Quaternion.identity);

}

}

gameField = new GameFieldPresentation(unit, FindObjectOfType<Settings>().gameObject, FindObjectOfType<Settings>().GameSettings.ChosenColorPlayer1.GetColor(), new Color(156F / 255F, 249F / 255F, 183F / 255F), 2);

gameField.Caterpillar.TurnsCountChanged += UpdateTurnsCount;

TurnsCount.GetComponent<Text>().text = gameField.Caterpillar.Turns.ToString();

GameObject.Find("Avatar").GetComponent<Image>().color = gameField.Caterpillar.Player1;

GameObject.FindGameObjectWithTag("Indicator").GetComponent<Animator>().SetTrigger("Show");

FindObjectOfType<ColorChange>().StartColorChange(gameField.Caterpillar.colorToSet);

}

void UpdateTurnsCount()

{

TurnsCount.GetComponent<Text>().text = gameField.Caterpillar.Turns.ToString();

}

}

## GooglePlayScript

using GooglePlayGames;

using GooglePlayGames.BasicApi;

using GooglePlayGames.BasicApi.Multiplayer;

using UnityEngine;

using UnityEngine.UI;

/// <summary>

/// Скрипт, отвечающий за взаимодействие с сервисами Google Play Games

/// </summary>

public class GooglePlayScript : MonoBehaviour {

bool displayNotification;

// Use this for initialization

void Start () {

// Create client configuration

PlayGamesClientConfiguration config = new PlayGamesClientConfiguration.Builder().WithInvitationDelegate(OnInvitationReceived).Build();

// Enable debugging output (recommended)

PlayGamesPlatform.DebugLogEnabled = true;

// Initialize and activate the platform

PlayGamesPlatform.InitializeInstance(config);

PlayGamesPlatform.Activate();

displayNotification = false;

PlayGamesPlatform.Instance.Authenticate(SignInCallback, true); //Вход во время запуска приложения

InitializeButton();

}

void SignInCallback(bool success)

{

if (success)

{

Debug.Log("(" + Social.localUser.userName + ") Signed in!");

InitializeButton();

PlayGamesPlatform.Instance.ReportProgress("CgkI35rhhvsWEAIQAQ", 100.0f, (bool s) => {

// handle success or failure

});

FindObjectOfType<Settings>().IncAchieve();

}

else

{

Debug.Log("(" + Social.localUser.userName + ") Sign-in failed...");

InitializeButton();

if (displayNotification)

{

FindObjectOfType<NotificationManager>().CantConnectPlayGames();

}

}

}

/// <summary>

/// Инициализирует кнопку хода\выхода в Play Games

/// </summary>

void InitializeButton()

{

if (PlayGamesPlatform.Instance.localUser.authenticated)

{

gameObject.GetComponentInChildren<Text>().text = "Disconnect Play Games";

}

if (!PlayGamesPlatform.Instance.localUser.authenticated)

{

gameObject.GetComponentInChildren<Text>().text = "Connect Play Games";

}

}

/// <summary>

/// Клик по кнопке входа\выхода в Play Games

/// </summary>

public void SettingsButtonClick()

{

if (!PlayGamesPlatform.Instance.localUser.authenticated)

{

displayNotification = true;

// Sign in with Play Game Services, showing the consent dialog

// by setting the second parameter to isSilent=false.

PlayGamesPlatform.Instance.Authenticate(SignInCallback, false);

}

else

{

// Sign out of play games

PlayGamesPlatform.Instance.SignOut();

InitializeButton();

}

}

/// <summary>

/// Клик по кнопке сетевой игры

/// </summary>

public void WebGameButtonClick()

{

if (!PlayGamesPlatform.Instance.localUser.authenticated)

{

FindObjectOfType<NotificationManager>().CantStartWebGame();

}

if (PlayGamesPlatform.Instance.localUser.authenticated)

{

WebGameManager.CreateWithInvitationScreen();

}

}

public void SignIn()

{

displayNotification = true;

if (!PlayGamesPlatform.Instance.localUser.authenticated)

{

// Sign in with Play Game Services, showing the consent dialog

// by setting the second parameter to isSilent=false.

PlayGamesPlatform.Instance.Authenticate(SignInCallback, false);

}

InitializeButton();

}

public void ShowAchievements()

{

PlayGamesPlatform.Instance.ShowAchievementsUI();

}

public void ShowInbox()

{

WebGameManager.AcceptFromInbox();

}

// called when an invitation is received:

public void OnInvitationReceived(Invitation invitation, bool shouldAutoAccept)

{

if (shouldAutoAccept)

{

// Invitation should be accepted immediately. This happens if the user already

// indicated (through the notification UI) that they wish to accept the invitation,

// so we should not prompt again.

WebGameManager.AcceptInvitation(invitation.InvitationId);

}

else

{

// The user has not yet indicated that they want to accept this invitation.

// We should \*not\* automatically accept it. Rather we store it and

// display an in-game popup:

NotificationManager.IncomingInvitation = invitation;

}

}

}

## InitializeCustomization

using UnityEngine;

using UnityEngine.UI;

using UnityEngine.EventSystems;

/// <summary>

/// Скрипт, инициализирующий элементы главного меню, отвечающие настройкам игры

/// </summary>

public class InitializeCustomization : MonoBehaviour {

public GameObject Default;

public GameObject Angry;

public GameObject Glamour;

public GameObject Music;

public GameObject Sound;

public GameObject DefColor;

public GameObject Color1;

public GameObject Color2;

// Use this for initialization

/// <summary>

/// Инициализирует элементы UI, отвечающие настройкам игры

/// </summary>

public void Initialize() {

print("initialization");

try

{

string skin = FindObjectOfType<Settings>().GameSettings.ChosenCaterpillar;

if (skin == "default\_caterpillar")

{

Default.GetComponent<EventTrigger>().OnPointerClick(new PointerEventData(FindObjectOfType<EventSystem>()));

}

if (skin == "angry\_caterpillar")

{

Angry.GetComponent<EventTrigger>().OnPointerClick(new PointerEventData(FindObjectOfType<EventSystem>()));

}

if (skin == "glamour\_caterpillar")

{

Glamour.GetComponent<EventTrigger>().OnPointerClick(new PointerEventData(FindObjectOfType<EventSystem>()));

}

string colortheme = FindObjectOfType<Settings>().GameSettings.ChosenColorTheme;

if (colortheme == "color\_0")

{

DefColor.GetComponent<EventTrigger>().OnPointerClick(new PointerEventData(FindObjectOfType<EventSystem>()));

}

if (colortheme == "color\_1")

{

Color1.GetComponent<EventTrigger>().OnPointerClick(new PointerEventData(FindObjectOfType<EventSystem>()));

}

if (colortheme == "color\_2")

{

Color2.GetComponent<EventTrigger>().OnPointerClick(new PointerEventData(FindObjectOfType<EventSystem>()));

}

Music.GetComponent<Toggle>().isOn = !FindObjectOfType<Settings>().GameSettings.MusicIsMute;

Sound.GetComponent<Toggle>().isOn = !FindObjectOfType<Settings>().GameSettings.SoundIsMute;

}

catch

{

}

}

void Start()

{

Initialize();

//вешаем звук на кпнопки

Button[] buttons = FindObjectsOfType<Button>();

foreach(Button button in buttons)

{

button.onClick.AddListener(() =>

{

FindObjectOfType<Settings>().sound.Play();

});

}

Toggle[] toggles = FindObjectsOfType<Toggle>();

foreach(Toggle toggle in toggles)

{

toggle.onValueChanged.AddListener((bool b) =>

{

FindObjectOfType<Settings>().sound.Play();

});

}

GameObject.Find("Color Panel").SetActive(false);

}

}

## InitializeGameSettings

using UnityEngine;

using UnityEngine.UI;

/// <summary>

/// Инициализирует элементы UI

/// </summary>

public class InitializeGameSettings : MonoBehaviour {

public GameObject Music;

public GameObject Sound;

// Use this for initialization

void Start () {

Music.GetComponent<Toggle>().isOn = !FindObjectOfType<Settings>().GameSettings.MusicIsMute;

Sound.GetComponent<Toggle>().isOn = !FindObjectOfType<Settings>().GameSettings.SoundIsMute;

Button[] buttons = FindObjectsOfType<Button>();

foreach (Button button in buttons)

{

button.onClick.AddListener(() =>

{

FindObjectOfType<Settings>().sound.Play();

});

}

Toggle[] toggles = FindObjectsOfType<Toggle>();

foreach (Toggle toggle in toggles)

{

toggle.onValueChanged.AddListener((bool b) =>

{

FindObjectOfType<Settings>().sound.Play();

});

}

}

}

## InitializeSet

using UnityEngine;

/// <summary>

/// Инстанцирует игровой объект настроек

/// </summary>

public class InitializeSet : MonoBehaviour {

public GameObject prefab;

// Use this for initialization

public void Start()

{

// инстанциировать игровой объект настроек, если его нет в текущей сцене

if (!GameObject.Find("Set"))

{

Instantiate(prefab);

}

}

}

## Link

using UnityEngine;

/// <summary>

/// Для перехода по ссылкам

/// </summary>

public class Link : MonoBehaviour {

public string url;

/// <summary>

/// Переход по ссылке

/// </summary>

public void GoToPage()

{

Application.OpenURL(url);

}

}

## Nod

using UnityEngine;

/// <summary>

/// Анимация гусеницы

/// </summary>

public class Nod : MonoBehaviour

{

//int intheLoop = 0;

Vector3 delta = new Vector3(0,1,0);

float speed = 0.01F;

Vector3 t = Vector3.up;

Vector3 pos;

// Use this for initialization

void Start ()

{

pos = transform.position;

}

// Update is called once per frame

void Update ()

{

transform.Translate(t \* Time.deltaTime \* speed);

if (System.Math.Abs(transform.position.y - pos.y) > delta.y)

ChangeMovementDirection();

}

void ChangeMovementDirection()

{

t = -t;

}

}

## NotificationManager

using UnityEngine;

using GooglePlayGames.BasicApi.Multiplayer;

/// <summary>

/// Управление уведомлениями

/// </summary>

public class NotificationManager : MonoBehaviour {

public static Invitation IncomingInvitation = null;

AndroidJavaObject context;

void ShowToastNotification(string toastString)

{

AndroidJavaClass unityPlayer = new AndroidJavaClass("com.unity3d.player.UnityPlayer");

AndroidJavaObject currentActivity = unityPlayer.GetStatic<AndroidJavaObject>("currentActivity");

context = currentActivity.Call<AndroidJavaObject>("getApplicationContext");

message = toastString;

currentActivity.Call("runOnUiThread", new AndroidJavaRunnable(showToast));

}

AndroidJavaObject toast;

AndroidJavaObject previousToast = null;

string message;

void showToast()

{

if (previousToast != null)

previousToast.Call("cancel");

AndroidJavaClass toastClass = new AndroidJavaClass("android.widget.Toast");

toast = toastClass.CallStatic<AndroidJavaObject>("makeText", context, message, toastClass.GetStatic<int>("LENGTH\_SHORT"));

toast.Call("show");

previousToast = toast;

}

// Update is called once per frame

void Update ()

{

if (IncomingInvitation != null)

{

NewInvitation();

IncomingInvitation = null;

}

}

public void CantConnectPlayGames()

{

ShowToastNotification("Connection to Play Games failed!\nTry again");

}

public void CantStartWebGame()

{

ShowToastNotification("For playing via the web you must\nbe connected to Play Games!");

}

public void WebGameSettingUpFailed()

{

ShowToastNotification("Can't start web game!\nTry again");

}

public void NewInvitation()

{

ShowToastNotification("You have new invitation to a match. Respond on it in the inbox!");

}

public void WebGameInterrupted(string message)

{

ShowToastNotification(message);

}

}

## PageSwitcher

using UnityEngine;

using UnityEngine.UI;

/// <summary>

/// Переключает страницы в главном меню

/// </summary>

public class PageSwitcher : MonoBehaviour {

public void PageSwitch()

{

if (gameObject.GetComponent<Text>().text == "Skins")

gameObject.GetComponent<Text>().text = "Colors";

else gameObject.GetComponent<Text>().text = "Skins";

}

}

## Panel

using UnityEngine;

/// <summary>

/// Управление панелями для реагирования на кнопку домой

/// </summary>

public class Panel : MonoBehaviour {

bool isOpen = false;

public GameObject PrevPanel;

bool focus = false;

bool postponedFocusChange = false;

void Start()

{

bool focus = false;

}

// Update is called once per frame

void Update () {

if (postponedFocusChange)

{

PrevPanel.GetComponent<Panel>().ChangeFocus(true);

postponedFocusChange = false;

}

if (!isOpen)

return;

if (!focus)

return;

if (Input.GetKeyDown(KeyCode.Escape))

{

Close();

}

}

public void Open()

{

OnPanelOpen();

gameObject.GetComponent<Animator>().SetTrigger("Appear");

if (PrevPanel == null)

GameObject.Find("Fade").GetComponent<Animator>().SetTrigger("Cover");

if (PrevPanel != null)

{

PrevPanel.GetComponent<Panel>().ChangeFocus(false);

}

}

public void Close()

{

isOpen = false;

ChangeFocus(false);

gameObject.GetComponent<Animator>().SetTrigger("Disappear");

if (PrevPanel == null)

GameObject.Find("Fade").GetComponent<Animator>().SetTrigger("Uncover");

if (PrevPanel != null)

{

PostponedFocusChange();

}

}

void PostponedFocusChange()

{

postponedFocusChange = true;

}

public void ChangeFocus(bool isInFocus)

{

focus = isInFocus;

}

void OnPanelOpen()

{

isOpen = true;

focus = true;

}

}

## Rotate

using UnityEngine;

/// <summary>

/// Анимация гусеницы

/// </summary>

public class Rotate : MonoBehaviour

{

Vector3 rotator = Vector3.forward;

public float speed = 25;

public int angle = 20;

// Update is called once per frame

void Update ()

{

if (angle != 0)

{

transform.Rotate(rotator \* Time.deltaTime \* speed);

if (transform.rotation.z \* 100 > angle | transform.rotation.z \* 100 < 0)

ChangeRotationDirection();

}

if (angle == 0)

{

transform.Rotate(Vector3.back \* Time.deltaTime \* speed);

}

}

void ChangeRotationDirection()

{

rotator = -rotator;

}

}

## Settings

using UnityEngine;

using System.IO;

using System.Runtime.Serialization.Formatters.Binary;

using GooglePlayGames;

/// <summary>

/// Класс, представляющий цвет

/// </summary>

[System.Serializable]

public class MyColor

{

// default 12, 239, 169

// angry 186, 85, 85

// glamour 206, 99, 196

public float r;

public float g;

public float b;

public MyColor(Color c)

{

r = c.r;

g = c.g;

b = c.b;

}

public MyColor(float r, float g, float b)

{

this.r = r;

this.g = g;

this.b = b;

}

public Color GetColor()

{

return new Color(r, g, b);

}

}

/// <summary>

/// Класс, хранящий настройки

/// </summary>

[System.Serializable]

public class SettingsHolder

{

public string ChosenCaterpillar;

public string ChosenColorTheme;

public MyColor ChosenColorPlayer1;

public MyColor ChosenColorPlayer2;

public bool MusicIsMute;

public bool SoundIsMute;

}

/// <summary>

/// Представляет настройки

/// </summary>

public class Settings : MonoBehaviour {

public SettingsHolder GameSettings;

public AudioSource sound;

public AudioSource fail;

public AudioSource music;

static bool incachieve = true;

void OnApplicationQuit()

{

incachieve = true;

}

public void IncAchieve()

{

if (incachieve)

{

print("incachieve");

incachieve = false;

PlayGamesPlatform.Instance.IncrementAchievement("CgkI35rhhvsWEAIQCA", 1,

(bool success) => {

// handle success or failure

});

}

}

void Start () {

Input.multiTouchEnabled = false;

DontDestroyOnLoad(gameObject);

gameObject.name = "Set";

if (File.Exists(Application.persistentDataPath + "/gamesettings.dat"))

{

LoadSettings();

}

else

{

GameSettings = new SettingsHolder();

GameSettings.ChosenCaterpillar = "default\_caterpillar";

GameSettings.ChosenColorTheme = "color\_0";

GameSettings.ChosenColorPlayer1 = new MyColor(12F/255F, 239F/255F, 169F/255F);

GameSettings.ChosenColorPlayer2 = new MyColor(40F/255F, 118F/255F, 234F/255F);

GameSettings.MusicIsMute = false;

GameSettings.SoundIsMute = false;

SaveSettings();

}

sound.mute = GameSettings.SoundIsMute;

fail.mute = GameSettings.SoundIsMute;

music.mute = GameSettings.MusicIsMute;

try

{

FindObjectOfType<InitializeCustomization>().Initialize();

}

catch { }

}

/// <summary>

/// Устанавливает новую цветовую тему и сохраняет выбор

/// </summary>

/// <param name="c1"></param>

/// <param name="c2"></param>

/// <param name="colorthemename"></param>

public void SetColor(Color c1, Color c2, string colorthemename)

{

GameSettings.ChosenColorPlayer1 = new MyColor(c1);

GameSettings.ChosenColorPlayer2 = new MyColor(c2);

GameSettings.ChosenColorTheme = colorthemename;

SaveSettings();

}

/// <summary>

/// Устанавливает новое название скина и сохраняет выбор

/// </summary>

/// <param name="name"> default\_caterpillar/angry\_caterpillar/glamour\_caterpillar</param>

public void SetCaterpillar(string name)

{

GameSettings.ChosenCaterpillar = name;

SaveSettings();

}

/// <summary>

/// Вкючает и выключает звук и сохраняет выбор

/// </summary>

public void SetSound()

{

GameSettings.SoundIsMute = !GameSettings.SoundIsMute;

sound.mute = GameSettings.SoundIsMute;

fail.mute = GameSettings.SoundIsMute;

SaveSettings();

}

/// <summary>

/// Вкючает и выключает музыку и сохраняет выбор

/// </summary>

public void SetMusic()

{

GameSettings.MusicIsMute = !GameSettings.MusicIsMute;

music.mute = GameSettings.MusicIsMute;

SaveSettings();

}

/// <summary>

/// Сериализует настройки

/// </summary>

void SaveSettings()

{

using (FileStream filestream = new FileStream(Application.persistentDataPath + "/gamesettings.dat", FileMode.Create))

{

BinaryFormatter formatter = new BinaryFormatter();

formatter.Serialize(filestream, GameSettings);

}

}

/// <summary>

/// Возвращает сериализованные настройки

/// </summary>

void LoadSettings()

{

using (FileStream filestream = new FileStream(Application.persistentDataPath + "/gamesettings.dat", FileMode.Open))

{

BinaryFormatter formatter = new BinaryFormatter();

GameSettings = (SettingsHolder) formatter.Deserialize(filestream);

}

}

}

## SkinsController

using UnityEngine;

/// <summary>

/// Кнопки скинов в меню

/// </summary>

public class SkinsController : MonoBehaviour {

public string nameToSet;

public GameObject Chosen;

/// <summary>

/// Устанавливает новый скин и обновляет UI в соответствии с новым выбором

/// </summary>

public void SetSkin()

{

Chosen.transform.position = gameObject.transform.position;

FindObjectOfType<Settings>().SetCaterpillar(nameToSet);

}

}

## Switcher

using UnityEngine;

/// <summary>

/// Включает и выключает объекты в меню кастомизации

/// </summary>

public class Switcher : MonoBehaviour {

/// <summary>

/// Включает и выключает объект, к которому прикреплен скрипт

/// </summary>

public void Switch()

{

gameObject.SetActive(!gameObject.activeSelf);

}

}

## Tutorial

using UnityEngine;

using UnityEngine.UI;

/// <summary>

/// Демонстрация правил игры в меню

/// </summary>

public class Tutorial : MonoBehaviour {

int clicks;

public GameObject Illustration;

public GameObject Subtitle;

/// <summary>

/// Инициализация при начале туториала

/// </summary>

public void StartTutorial()

{

clicks = 0;

Illustration.SetActive(true);

Illustration.GetComponent<Image>().sprite = Resources.Load<Sprite>("Menu/tutorial" + clicks % 8);

gameObject.SetActive(true);

Subtitle.SetActive(true);

Subtitle.GetComponent<Text>().text = "Rules";

GameObject.Find("Instructions").GetComponent<Text>().text = "Add new parts to the caterpillar by turns with your partner to make it longer. On each turn by default you have three parts to place. On the first turn new part must be added to the head of the caterpillar, then to the the highlighted part which has been added last.";

}

/// <summary>

/// Пролистывание страниц туториала

/// </summary>

public void OnButtonClickHandler()

{

clicks++;

if (clicks < 11)

{

Illustration.GetComponent<Image>().sprite = Resources.Load<Sprite>("Menu/tutorial" + clicks);

if (clicks == 4)

GameObject.Find("Instructions").GetComponent<Text>().text = "Game ends when one of the players got in a deadlock.";

if (clicks == 6)

GameObject.Find("Instructions").GetComponent<Text>().text = "Game is over: it's turn of the green player, but he can't add any new part to the end of the caterpillar. Blue player has won.";

if (clicks == 7)

GameObject.Find("Instructions").GetComponent<Text>().text = "Apples give opportunity to make two additional moves to the player who placed an element on its cell.";

if (clicks == 8)

{

Subtitle.GetComponent<Text>().text = "Tips";

GameObject.Find("Instructions").GetComponent<Text>().text = "For playing via the web and unlocking achievements you should sign in Play Games with your Google account.";

}

if (clicks == 9)

GameObject.Find("Instructions").GetComponent<Text>().text = "To improve your game expirience make sure that your internet connection is stable before starting a web match. Otherwise the match may intermit.";

if (clicks == 10)

GameObject.Find("Instructions").GetComponent<Text>().text = "Do not minimize the app while you are playing a web game or Play Services will interrupt it.";

}

else

{

Illustration.SetActive(false);

Subtitle.SetActive(false);

gameObject.SetActive(false);

GameObject.Find("Instructions").GetComponent<Text>().text = "Have a nice game!";

}

}

}

## UnitController

using GooglePlayGames;

using UnityEngine;

using UnityEngine.UI;

/// <summary>

/// Скрипт ячейки поля во время игры против компьютера

/// </summary>

public class UnitController : MonoBehaviour {

public int x;

public int y;

/// <summary>

/// Для игрока

/// </summary>

public void AttemptToPlaceElement()

{

if (FindObjectOfType<GameVsComputerController>().gameField.PlayerOnMove == 1)

if (!FindObjectOfType<GameVsComputerController>().gameField.ChangingPlayer)

{

if (gameObject.tag == "Empty" || gameObject.tag == "Food")

{

if (FindObjectOfType<GameVsComputerController>().gameField.Caterpillar.UnitIsOkToAdd(x, y) && FindObjectOfType<GameVsComputerController>().gameField.TouchOnlyOneBodyElement(x, y))

{

FindObjectOfType<GameVsComputerController>().gameField.PlaySound();

FindObjectOfType<GameVsComputerController>().Turns++;

if (FindObjectOfType<GameVsComputerController>().Turns == 9)

{

PlayGamesPlatform.Instance.ReportProgress("CgkI35rhhvsWEAIQBQ", 100.0f, (bool s) => {

// handle success or failure

});

}

if (gameObject.tag == "Food")

{

FindObjectOfType<GameVsComputerController>().gameField.Caterpillar.Turns += 2;

FindObjectOfType<GameVsComputerController>().Apples++;

if (FindObjectOfType<GameVsComputerController>().Apples == 5)

PlayGamesPlatform.Instance.ReportProgress("CgkI35rhhvsWEAIQBA", 100.0f, (bool s) => {

// handle success or failure

});

}

gameObject.tag = "BodyElement";

transform.GetChild(3).gameObject.SetActive(true);

FindObjectOfType<GameVsComputerController>().gameField.Caterpillar.Turns--;

transform.GetChild(0).GetComponent<SpriteRenderer>().color = FindObjectOfType<GameVsComputerController>().gameField.Caterpillar.colorToSet;

transform.GetChild(4).gameObject.SetActive(false);

FindObjectOfType<Bot>().PrevX = FindObjectOfType<GameVsComputerController>().gameField.Caterpillar.ActiveX;

FindObjectOfType<Bot>().PrevY = FindObjectOfType<GameVsComputerController>().gameField.Caterpillar.ActiveY;

FindObjectOfType<GameVsComputerController>().gameField.Caterpillar.SetActiveUnit(x, y);

if (FindObjectOfType<GameVsComputerController>().gameField.Caterpillar.Turns == 0)

{

FindObjectOfType<GameVsComputerController>().Turns = 0;

FindObjectOfType<GameVsComputerController>().gameField.ChangingPlayer = true;

GameObject.FindGameObjectWithTag("Indicator").GetComponent<Animator>().SetTrigger("Change");

if (FindObjectOfType<GameVsComputerController>().gameField.PlayerOnMove == 1)

FindObjectOfType<GameVsComputerController>().gameField.Caterpillar.colorToSet = FindObjectOfType<GameVsComputerController>().gameField.Caterpillar.Player2;

else FindObjectOfType<GameVsComputerController>().gameField.Caterpillar.colorToSet = FindObjectOfType<GameVsComputerController>().gameField.Caterpillar.Player1;

FindObjectOfType<ColorChange>().StartColorChange(FindObjectOfType<GameVsComputerController>().gameField.Caterpillar.colorToSet);

if (FindObjectOfType<GameVsComputerController>().PlayerOnMove == 1)

FindObjectOfType<GameVsComputerController>().PlayerOnMove = 2;

else FindObjectOfType<GameVsComputerController>().PlayerOnMove = 1;

}

if (!FindObjectOfType<GameVsComputerController>().gameField.AvailableMoves())

{

if (FindObjectOfType<GameVsComputerController>().gameField.PlayerOnMove == 1)

{

GameObject.Find("Winner").GetComponent<Text>().text = "You lose";

}

else

{

PlayGamesPlatform.Instance.IncrementAchievement("CgkI35rhhvsWEAIQAw", 1,

(bool success) => {

// handle success or failure

});

GameObject.Find("Winner").GetComponent<Text>().text = "You won";

}

GameObject.Find("Fade").GetComponent<Animator>().SetTrigger("Cover");

GameObject.Find("Game Over").GetComponent<Animator>().SetTrigger("Appear");

}

return;

}

FindObjectOfType<GameVsComputerController>().gameField.PlayFail();

}

}

}

/// <summary>

/// Для компьютера

/// </summary>

public void PlaceElementForBot()

{

if (!FindObjectOfType<GameVsComputerController>().gameField.ChangingPlayer)

{

if (gameObject.tag == "Empty" || gameObject.tag == "Food")

{

if (FindObjectOfType<GameVsComputerController>().gameField.Caterpillar.UnitIsOkToAdd(x, y) && FindObjectOfType<GameVsComputerController>().gameField.TouchOnlyOneBodyElement(x, y))

{

//print("set");

FindObjectOfType<GameVsComputerController>().gameField.PlaySound();

if (gameObject.tag == "Food")

FindObjectOfType<GameVsComputerController>().gameField.Caterpillar.Turns += 2;

gameObject.tag = "BodyElement";

transform.GetChild(3).gameObject.SetActive(true);

FindObjectOfType<GameVsComputerController>().gameField.Caterpillar.Turns--;

transform.GetChild(0).GetComponent<SpriteRenderer>().color = FindObjectOfType<GameVsComputerController>().gameField.Caterpillar.colorToSet;

transform.GetChild(4).gameObject.SetActive(false);

FindObjectOfType<Bot>().PrevX = FindObjectOfType<GameVsComputerController>().gameField.Caterpillar.ActiveX;

FindObjectOfType<Bot>().PrevY = FindObjectOfType<GameVsComputerController>().gameField.Caterpillar.ActiveY;

FindObjectOfType<GameVsComputerController>().gameField.Caterpillar.SetActiveUnit(x, y);

if (FindObjectOfType<GameVsComputerController>().gameField.Caterpillar.Turns == 0)

{

FindObjectOfType<GameVsComputerController>().Turns = 0;

FindObjectOfType<GameVsComputerController>().gameField.ChangingPlayer = true;

GameObject.FindGameObjectWithTag("Indicator").GetComponent<Animator>().SetTrigger("Change");

if (FindObjectOfType<GameVsComputerController>().gameField.PlayerOnMove == 1)

FindObjectOfType<GameVsComputerController>().gameField.Caterpillar.colorToSet = FindObjectOfType<GameVsComputerController>().gameField.Caterpillar.Player2;

else FindObjectOfType<GameVsComputerController>().gameField.Caterpillar.colorToSet = FindObjectOfType<GameVsComputerController>().gameField.Caterpillar.Player1;

FindObjectOfType<ColorChange>().StartColorChange(FindObjectOfType<GameVsComputerController>().gameField.Caterpillar.colorToSet);

if (FindObjectOfType<GameVsComputerController>().PlayerOnMove == 1)

FindObjectOfType<GameVsComputerController>().PlayerOnMove = 2;

else FindObjectOfType<GameVsComputerController>().PlayerOnMove = 1;

}

if (!FindObjectOfType<GameVsComputerController>().gameField.AvailableMoves())

{

if (FindObjectOfType<GameVsComputerController>().gameField.PlayerOnMove == 1)

{

GameObject.Find("Winner").GetComponent<Text>().text = "You lose";

}

else

{

PlayGamesPlatform.Instance.IncrementAchievement("CgkI35rhhvsWEAIQAw", 1,

(bool success) => {

// handle success or failure

});

GameObject.Find("Winner").GetComponent<Text>().text = "You won";

}

GameObject.Find("Fade").GetComponent<Animator>().SetTrigger("Cover");

GameObject.Find("Game Over").GetComponent<Animator>().SetTrigger("Appear");

}

}

}

}

}

}

## UnitControllerWeb

using GooglePlayGames;

using UnityEngine;

using UnityEngine.UI;

/// <summary>

/// Скрипт ячейки поля во время игры по сети

/// </summary>

public class UnitControllerWeb : MonoBehaviour {

public int x;

public int y;

/// <summary>

/// Для игрока на этом устройстве

/// </summary>

public void AttemptToPlaceElement()

{

if (FindObjectOfType<WebGameController>().gameField.PlayerOnMove == WebGameManager.Instance.MyPlayerNumber)

if (!FindObjectOfType<WebGameController>().gameField.ChangingPlayer)

{

if (gameObject.tag == "Empty" || gameObject.tag == "Food")

{

if (FindObjectOfType<WebGameController>().gameField.Caterpillar.UnitIsOkToAdd(x, y) && FindObjectOfType<WebGameController>().gameField.TouchOnlyOneBodyElement(x, y))

{

FindObjectOfType<WebGameController>().gameField.PlaySound();

FindObjectOfType<WebGameController>().SendTurn(x, y); //отправка хода оппоненту

if (gameObject.tag == "Food")

{

FindObjectOfType<WebGameController>().gameField.Caterpillar.Turns += 2;

}

gameObject.tag = "BodyElement";

transform.GetChild(3).gameObject.SetActive(true);

FindObjectOfType<WebGameController>().gameField.Caterpillar.Turns--;

transform.GetChild(0).GetComponent<SpriteRenderer>().color = FindObjectOfType<WebGameController>().gameField.Caterpillar.colorToSet;

transform.GetChild(4).gameObject.SetActive(false);

FindObjectOfType<WebGameController>().gameField.Caterpillar.SetActiveUnit(x, y);

if (FindObjectOfType<WebGameController>().gameField.Caterpillar.Turns == 0)

{

FindObjectOfType<WebGameController>().gameField.ChangingPlayer = true;

GameObject.FindGameObjectWithTag("Indicator").GetComponent<Animator>().SetTrigger("Change");

if (FindObjectOfType<WebGameController>().gameField.PlayerOnMove == 1)

FindObjectOfType<WebGameController>().gameField.Caterpillar.colorToSet = FindObjectOfType<WebGameController>().gameField.Caterpillar.Player2;

else FindObjectOfType<WebGameController>().gameField.Caterpillar.colorToSet = FindObjectOfType<WebGameController>().gameField.Caterpillar.Player1;

FindObjectOfType<ColorChange>().StartColorChange(FindObjectOfType<WebGameController>().gameField.Caterpillar.colorToSet);

if (FindObjectOfType<WebGameController>().gameField.PlayerOnMove == 1)

FindObjectOfType<WebGameController>().gameField.PlayerOnMove = 2;

else FindObjectOfType<WebGameController>().gameField.PlayerOnMove = 1;

}

if (!FindObjectOfType<WebGameController>().gameField.AvailableMoves())

{

PlayGamesPlatform.Instance.IncrementAchievement("CgkI35rhhvsWEAIQBw", 1,

(bool success) => {

// handle success or failure

});

WebGameManager.Instance.State = WebGameManager.GameState.Finished;

if (FindObjectOfType<WebGameController>().gameField.PlayerOnMove == WebGameManager.Instance.MyPlayerNumber)

{

GameObject.Find("Winner").GetComponent<Text>().text = "You lose";

}

else

{

GameObject.Find("Winner").GetComponent<Text>().text = "You won";

}

GameObject.Find("Fade").GetComponent<Animator>().SetTrigger("Cover");

GameObject.Find("Game Over").GetComponent<Animator>().SetTrigger("Appear");

}

return;

}

FindObjectOfType<WebGameController>().gameField.PlayFail();

}

}

}

/// <summary>

/// Для ходов оппонента

/// </summary>

public void PlaceElement()

{

if (gameObject.tag == "Empty" || gameObject.tag == "Food")

{

if (FindObjectOfType<WebGameController>().gameField.Caterpillar.UnitIsOkToAdd(x, y) && FindObjectOfType<WebGameController>().gameField.TouchOnlyOneBodyElement(x, y))

{

FindObjectOfType<WebGameController>().gameField.PlaySound();

if (gameObject.tag == "Food")

FindObjectOfType<WebGameController>().gameField.Caterpillar.Turns += 2;

gameObject.tag = "BodyElement";

transform.GetChild(3).gameObject.SetActive(true);

FindObjectOfType<WebGameController>().gameField.Caterpillar.Turns--;

transform.GetChild(0).GetComponent<SpriteRenderer>().color = FindObjectOfType<WebGameController>().gameField.Caterpillar.colorToSet;

transform.GetChild(4).gameObject.SetActive(false);

FindObjectOfType<WebGameController>().gameField.Caterpillar.SetActiveUnit(x, y);

if (FindObjectOfType<WebGameController>().gameField.Caterpillar.Turns == 0)

{

FindObjectOfType<WebGameController>().gameField.ChangingPlayer = true;

GameObject.FindGameObjectWithTag("Indicator").GetComponent<Animator>().SetTrigger("Change");

if (FindObjectOfType<WebGameController>().gameField.PlayerOnMove == 1)

FindObjectOfType<WebGameController>().gameField.Caterpillar.colorToSet = FindObjectOfType<WebGameController>().gameField.Caterpillar.Player2;

else FindObjectOfType<WebGameController>().gameField.Caterpillar.colorToSet = FindObjectOfType<WebGameController>().gameField.Caterpillar.Player1;

FindObjectOfType<ColorChange>().StartColorChange(FindObjectOfType<WebGameController>().gameField.Caterpillar.colorToSet);

if (FindObjectOfType<WebGameController>().gameField.PlayerOnMove == 1)

FindObjectOfType<WebGameController>().gameField.PlayerOnMove = 2;

else FindObjectOfType<WebGameController>().gameField.PlayerOnMove = 1;

}

if (!FindObjectOfType<WebGameController>().gameField.AvailableMoves())

{

PlayGamesPlatform.Instance.IncrementAchievement("CgkI35rhhvsWEAIQBw", 1,

(bool success) => {

// handle success or failure

});

WebGameManager.Instance.State = WebGameManager.GameState.Finished;

if (FindObjectOfType<WebGameController>().gameField.PlayerOnMove == WebGameManager.Instance.MyPlayerNumber)

{

GameObject.Find("Winner").GetComponent<Text>().text = "You lose";

}

else

{

GameObject.Find("Winner").GetComponent<Text>().text = "You won";

}

GameObject.Find("Fade").GetComponent<Animator>().SetTrigger("Cover");

GameObject.Find("Game Over").GetComponent<Animator>().SetTrigger("Appear");

}

}

}

}

}

## WebGameController

using GooglePlayGames;

using UnityEngine;

using UnityEngine.UI;

/// <summary>

/// Скрипт для игры по сети

/// </summary>

public class WebGameController : MonoBehaviour {

GameObject[,] unit;

public GameFieldPresentation gameField;

public GameObject prefab;

public GameObject TurnsCount;

// Use this for initialization

void Start () {

unit = new GameObject[15, 10];

for (int i = 0; i < 15; i++)

{

for (int j = 0; j < 10; j++)

{

unit[i, j] = Instantiate(prefab, new Vector3(-2.25F + 0.5F \* j, 3F - 0.5F \* i, 0), Quaternion.identity);

}

}

}

/// <summary>

/// Первоначальное создание игрового поля

/// </summary>

public void CreateGameField(Color mycolor, Color opponentscolor)

{

gameField = new GameFieldPresentation(unit, FindObjectOfType<Settings>().gameObject, mycolor, opponentscolor);

gameField.Caterpillar.TurnsCountChanged += UpdateTurnsCount;

TurnsCount.GetComponent<Text>().text = gameField.Caterpillar.Turns.ToString();

GameObject.Find("Avatar").GetComponent<Image>().color = gameField.Caterpillar.colorToSet;

GameObject.FindGameObjectWithTag("Indicator").GetComponent<Animator>().SetTrigger("Show");

FindObjectOfType<ColorChange>().StartColorChange(gameField.Caterpillar.colorToSet);

}

/// <summary>

/// Воссоздание игрового поля по переданным данным

/// </summary>

/// <param name="arr"></param>

/// <param name="mycolor"></param>

/// <param name="opponentscolor"></param>

public void InitializeGameField(byte[] arr, Color mycolor, Color opponentscolor)

{

gameField = new GameFieldPresentation(unit, FindObjectOfType<Settings>().gameObject, opponentscolor, mycolor, arr);

gameField.Caterpillar.TurnsCountChanged += UpdateTurnsCount;

TurnsCount.GetComponent<Text>().text = gameField.Caterpillar.Turns.ToString();

GameObject.Find("Avatar").GetComponent<Image>().color = gameField.Caterpillar.colorToSet;

GameObject.FindGameObjectWithTag("Indicator").GetComponent<Animator>().SetTrigger("Show");

FindObjectOfType<ColorChange>().StartColorChange(gameField.Caterpillar.colorToSet);

}

/// <summary>

/// Отправка сделанного хода оппоненту

/// </summary>

/// <param name="x"></param>

/// <param name="y"></param>

public void SendTurn(int x, int y)

{

byte[] turn = new byte[3];

turn[0] = (byte)'T';

turn[1] = (byte)x;

turn[2] = (byte)y;

PlayGamesPlatform.Instance.RealTime.SendMessageToAll(true, turn);

}

/// <summary>

/// Совершение полученного хода

/// </summary>

/// <param name="x"></param>

/// <param name="y"></param>

public void PlaceElement(int x, int y)

{

gameField.unit[x, y].GetComponent<UnitControllerWeb>().PlaceElement();

}

void UpdateTurnsCount()

{

TurnsCount.GetComponent<Text>().text = gameField.Caterpillar.Turns.ToString();

}

}

## WebGameManager

using UnityEngine;

using GooglePlayGames;

using GooglePlayGames.BasicApi.Multiplayer;

using System.Collections.Generic;

using UnityEngine.SceneManagement;

/// <summary>

/// Класс для управления сетевой игрой

/// </summary>

public class WebGameManager : RealTimeMultiplayerListener

{

static WebGameManager sInstance = null;

public delegate void Delegate(GameState state);

public enum GameState

{

SettingUp,

Loading,

Playing,

Finished,

SetupFailed,

Aborted

};

GameState mGameState = GameState.SettingUp;

public int MyPlayerNumber;

public Color MyColor;

public Color OpponentsColor;

public static event Delegate GameStateChanged;

/// <summary>

/// Возвращает экземпляр класса

/// </summary>

public static WebGameManager Instance

{

get

{

return sInstance;

}

}

/// <summary>

/// Состояние игры

/// </summary>

public GameState State

{

get

{

return mGameState;

}

set

{

mGameState = value;

if (GameStateChanged != null)

GameStateChanged(mGameState);

}

}

public static void CreateWithInvitationScreen()

{

sInstance = new WebGameManager();

PlayGamesPlatform.Instance.RealTime.CreateWithInvitationScreen(1, 1, 0, sInstance);

}

public static void AcceptFromInbox()

{

sInstance = new WebGameManager();

PlayGamesPlatform.Instance.RealTime.AcceptFromInbox(sInstance);

}

public static void AcceptInvitation(string invitationID)

{

sInstance = new WebGameManager();

PlayGamesPlatform.Instance.RealTime.AcceptInvitation(invitationID, sInstance);

}

public static void LeaveGame()

{

GameStateChanged -= GameObject.Find("Game Panel").GetComponent<WebGameUI>().GameStateChangedHandler;

SceneManager.LoadScene("menu");

if (PlayGamesPlatform.Instance.RealTime.IsRoomConnected())

{

PlayGamesPlatform.Instance.RealTime.LeaveRoom();

}

}

/// <summary>

/// Когда игрок подключается к сетевому матчу

/// </summary>

/// <param name="success"></param>

public void OnRoomConnected(bool success)

{

if (success)

{

State = GameState.Loading;

SceneManager.LoadScene("gameviatheweb");

//отправить сообщение о моем выборе цвета

byte[] reportMyColor;

reportMyColor = new byte[2];

reportMyColor[0] = (byte)'C';

try

{

reportMyColor[1] = (byte)GameObject.Find("Set").GetComponent<Settings>().GameSettings.ChosenColorTheme[GameObject.Find("Set").GetComponent<Settings>().GameSettings.ChosenColorTheme.Length - 1];

}

catch

{

reportMyColor[1] = 0;

}

PlayGamesPlatform.Instance.RealTime.SendMessageToAll(true, reportMyColor);

}

else

{

State = GameState.SetupFailed;

GameObject.Find("Notification").GetComponent<NotificationManager>().WebGameSettingUpFailed();

}

}

public void OnLeftRoom()

{

sInstance = null;

}

private bool showingWaitingRoom = false;

public void OnRoomSetupProgress(float progress)

{

// show the default waiting room.

if (!showingWaitingRoom)

{

showingWaitingRoom = true;

PlayGamesPlatform.Instance.RealTime.ShowWaitingRoomUI();

}

}

public void OnParticipantLeft(Participant participant)

{

if (State != GameState.Finished)

{

State = GameState.Aborted;

GameObject.Find("Notification").GetComponent<NotificationManager>().WebGameInterrupted("Your opponent has left the match.");

}

}

public void OnPeersConnected(string[] participantIds)

{

}

public void OnPeersDisconnected(string[] participantIds)

{

if (State != GameState.Finished)

{

State = GameState.Aborted;

GameObject.Find("Notification").GetComponent<NotificationManager>().WebGameInterrupted("Your opponent has been disconnected from the match.");

}

}

/// <summary>

/// Определяет мой номер как игрока на основе списка участников матча

/// </summary>

public void SetMyNumber()

{

List<Participant> participants = PlayGamesPlatform.Instance.RealTime.GetConnectedParticipants();

for (int i = 0; i < participants.Count; i++)

{

if (participants[i].Equals(PlayGamesPlatform.Instance.RealTime.GetSelf()))

{

MyPlayerNumber = i + 1;

break;

}

}

}

public void OnRealTimeMessageReceived(bool isReliable, string senderId, byte[] data)

{

switch ((char) data[0])

{

case 'C': //сообщение о цвете

//определить мой номер как игрока

SetMyNumber();

char mycolorNumber = GameObject.Find("Set").GetComponent<Settings>().GameSettings.ChosenColorTheme[GameObject.Find("Set").GetComponent<Settings>().GameSettings.ChosenColorTheme.Length - 1];

if (mycolorNumber == (char) data[1])

{

if (MyPlayerNumber == 1)

{

MyColor = GameObject.Find("Set").GetComponent<Settings>().GameSettings.ChosenColorPlayer1.GetColor();

OpponentsColor = GameObject.Find("Set").GetComponent<Settings>().GameSettings.ChosenColorPlayer2.GetColor();

}

if (MyPlayerNumber == 2)

{

MyColor = GameObject.Find("Set").GetComponent<Settings>().GameSettings.ChosenColorPlayer2.GetColor();

OpponentsColor = GameObject.Find("Set").GetComponent<Settings>().GameSettings.ChosenColorPlayer1.GetColor();

}

}

else

{

MyColor = GameObject.Find("Set").GetComponent<Settings>().GameSettings.ChosenColorPlayer1.GetColor();

switch ((char)data[1])

{

case '0':

OpponentsColor = new MyColor(12F / 255F, 239F / 255F, 169F / 255F).GetColor();

break;

case '1':

OpponentsColor = new MyColor(186F / 255F, 85F / 255F, 85F / 255F).GetColor();

break;

case '2':

OpponentsColor = new MyColor(206F / 255F, 99F / 255F, 196F / 255F).GetColor();

break;

}

}

State = GameState.Playing;

//первый игрок инициализирует игровое поле у себя и делится информацией

if (MyPlayerNumber == 1)

{

GameObject.Find("Control").GetComponent<WebGameController>().CreateGameField(MyColor, OpponentsColor);

byte[] config = GameObject.Find("Control").GetComponent<WebGameController>().gameField.GetFieldConfig();

PlayGamesPlatform.Instance.RealTime.SendMessageToAll(true, config);

}

break;

case 'I': //сообщение конфигурации игрового поля

GameObject.Find("Control").GetComponent<WebGameController>().InitializeGameField(data, MyColor, OpponentsColor);

break;

case 'T': //сообщение о ходе оппонента

GameObject.Find("Control").GetComponent<WebGameController>().PlaceElement(data[1], data[2]);

break;

}

}

}

## WebGameUI

using UnityEngine;

using UnityEngine.UI;

/// <summary>

/// Управление загрузочным экраном

/// </summary>

public class WebGameUI : MonoBehaviour {

public GameObject GamePanel;

public GameObject Loading;

public void Start()

{

//добавление обработчика события

WebGameManager.GameStateChanged += GameStateChangedHandler;

}

public void GameStateChangedHandler(WebGameManager.GameState state)

{

switch (state)

{

case WebGameManager.GameState.SetupFailed:

GamePanel.GetComponentInChildren<Text>().text = "Setup failed.";

Loading.SetActive(false);

break;

case WebGameManager.GameState.Loading:

GamePanel.GetComponentInChildren<Text>().text = "Loading!";

Loading.SetActive(true);

break;

case WebGameManager.GameState.Playing:

//GamePanel.GetComponentInChildren<Text>().text = "Playing!";

if (WebGameManager.Instance.MyPlayerNumber == 1)

GamePanel.GetComponentInChildren<Text>().text = "First turn is yours!";

else GamePanel.GetComponentInChildren<Text>().text = "Your opponent starts!";

Loading.SetActive(false);

GamePanel.GetComponent<Animator>().SetTrigger("Start");

break;

case WebGameManager.GameState.Aborted:

Loading.SetActive(false);

GamePanel.GetComponentInChildren<Text>().text = "Something gone wrong";

GamePanel.GetComponent<Animator>().SetTrigger("Appear");

break;

case WebGameManager.GameState.Finished:

break;

}

}

public void LeaveGame()

{

WebGameManager.LeaveGame();

}

}

# ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Номера листов (страниц) | | | | | Всего листов (страниц в докум.) | № документа | Входящий № сопроводительного докум. и дата | Подп. | Дата |
| Изм. | Измененных | Замененных | Новых | Аннулированх |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |