



DOKUMENTACJA PROGRAMISTYCZNA

ALEKSANDER CHYTKIEWICZ

WYDZIAŁ ELEKTRYCZNY, INFORMATYKA

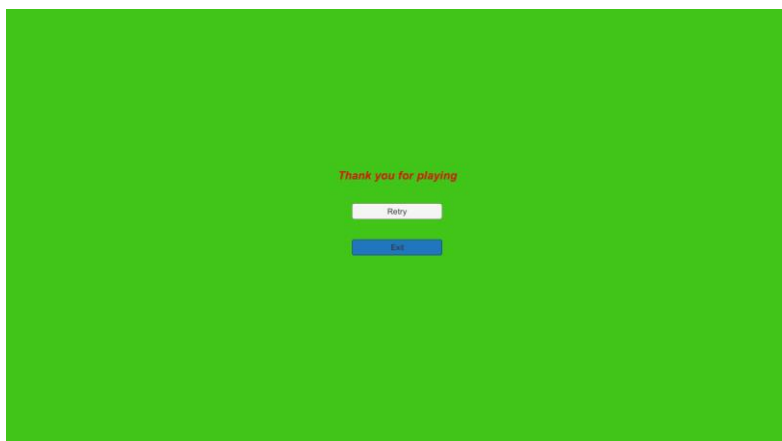
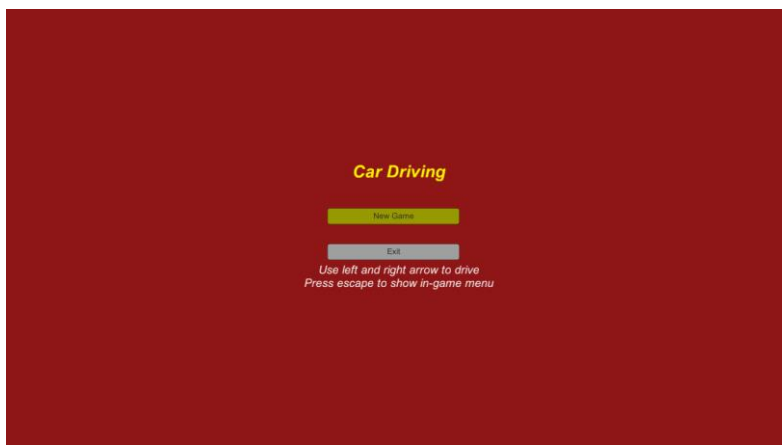
NICK NA CODEGURU: REPAY

NAZWA PROGRAMU (GRY): CAR DRIVING

Krótki opis gry

Jest to gra 2D, polegająca na przejechaniu przez wyboisty teren i dotarciu do mety (oznaczonej jako flaga). Gracz porusza się pojazdem za pomocą strzałek (w prawo i w lewo). Pojazdem można także sterować w powietrzu (również poprzez strzałki). Przycisk Escape wywołuje menu w grze. W menu możemy wybrać 3 opcje (powrót do gry, rozpoczęcie od nowa oraz wyjście z gry). Uruchomienie gry wyświetla proste menu, natomiast po dotarciu do mety mamy inne menu, które pozwala nam powtórzyć grę lub wyłączyć ją.

Zrzuty ekranu

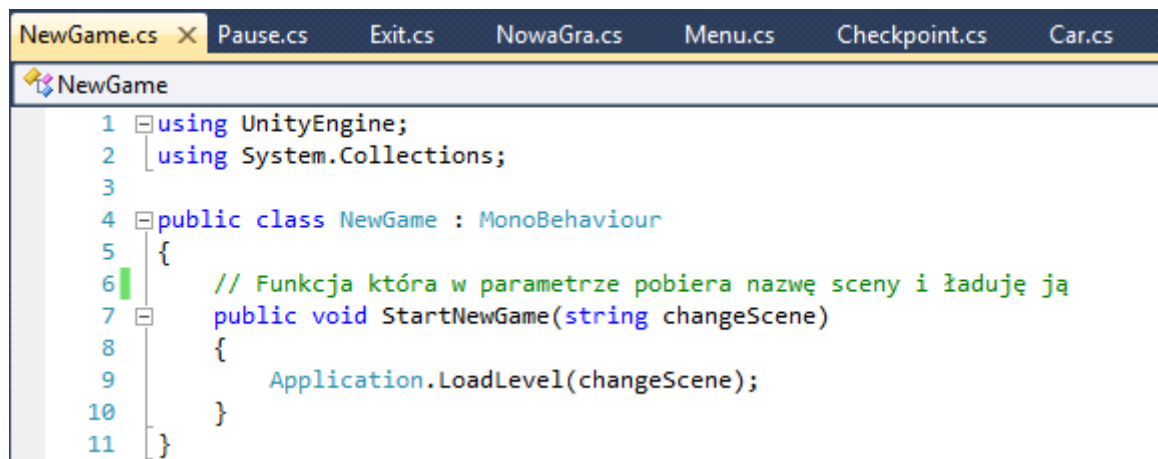


Kod źródłowy

W programie wykorzystano 6 skryptów, napisanych w języku C# oraz stworzono 3 sceny. Wykorzystano głównie elementy Canvas'u (tekst i przyciski) oraz proste bloki.

Skrypt „NewGame”

Wykorzystany w przyciskach. Służy do załadowania właściwej sceny z grą, po naciśnięciu przycisku „New Game” albo „Retry”.

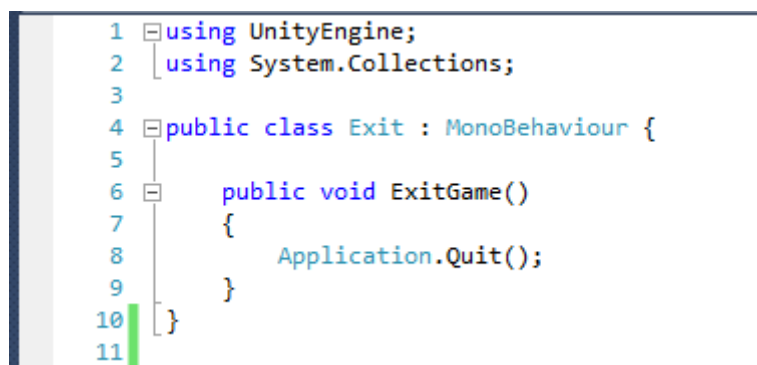


The screenshot shows the Unity IDE with several script files open in the top bar: NewGame.cs, Pause.cs, Exit.cs, NowaGra.cs, Menu.cs, Checkpoint.cs, and Car.cs. The NewGame.cs script is selected and its code is visible in the editor. The code defines a MonoBehaviour class named NewGame with a StartNewGame method that calls Application.LoadLevel.

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class NewGame : MonoBehaviour
5 {
6     // Funkcja która w parametrze pobiera nazwę sceny i ładuje ją
7     public void StartNewGame(string changeScene)
8     {
9         Application.LoadLevel(changeScene);
10    }
11 }
```

Skrypt „Exit”

Wykorzystywany w przyciskach. Służy do wyłączenia programu po naciśnięciu przyciska „Exit”.



The screenshot shows the Unity IDE with the Exit.cs script selected and its code visible in the editor. The code defines a MonoBehaviour class named Exit with an ExitGame method that calls Application.Quit.

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class Exit : MonoBehaviour {
5
6     public void ExitGame()
7     {
8         Application.Quit();
9     }
10 }
11
```

Skrypt „Checkpoint”

Wykorzystany w obiekcie flaga, który sprawdza kolizję obiektu z pojazdem. Gdy do niej dojdzie, ładuje scenę końcową, charakterystyczną dla ukończenia gry.

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class Checkpoint : MonoBehaviour
5 {
6
7     // Służy do załadowania Levelu End, po kolizji z pojazdem
8     void OnCollisionEnter2D(Collision2D other)
9     {
10         if (other.gameObject.name == "Car")
11         {
12             Application.LoadLevel("End");
13         }
14     }
15 }
16
```

Skrypt „Pause”

Służy do wywołania menu w grze.

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class Pause : MonoBehaviour {
5
6     // Inicjalizacja zmiennych
7     private bool showMenu = false;
8     private const int INGAME_MENU_WINDOW_ID = 0;
9     private Rect ingameWindowRect = new Rect(Screen.width / 2 - 85, Screen.height / 2 - 200, 170, 200);
10    public GUISkin gameSkin;
11
12    void Start()
13    {
14        Screen.showCursor = false;
15        Screen.lockCursor = true;
16    }
17
18    // Po naciśnięciu Escape ma pokazać menu
19    void Update()
20    {
21        if (Input.GetKeyDown(KeyCode.Escape))
22        {
23            showMenu = !showMenu;
24        }
25    }
26 }
```

```

27 // Funkcja obsługująca zdarzenia w GUI
28 void OnGUI()
29 {
30     GUI.skin = gameSkin;
31
32     if (showMenu)
33     {
34         ingameWindowRect = GUI.Window(INGAME_MENU_WINDOW_ID, ingameWindowRect, IngameMenuDisplay, "");
35
36         Screen.showCursor = true;
37         Screen.lockCursor = false;
38
39         if (showMenu == false)
40         {
41             Screen.showCursor = false;
42             Screen.lockCursor = true;
43         }
44     }
45 }

```

```

47 // Funkcja tworząca kolejne przyciski i opisująca ich akcję po naciśnięciu
48 public void IngameMenuDisplay(int INGAME_MENU_WINDOW_ID)
49 {
50     if (GUI.Button(new Rect(10, 30, 150, 32), "Return", "Button"))
51     {
52         showMenu = false;
53     }
54
55     if (GUI.Button(new Rect(10, 70, 150, 32), "Retry", "Button"))
56     {
57         Application.LoadLevel("mission1");
58     }
59
60     if (GUI.Button(new Rect(10, 110, 150, 32), "Exit", "Button"))
61     {
62         Application.Quit();
63     }
64 }

```

Skrypt „FallowCamera”

Dzięki temu kamera podąża za naszym pojazdem.

```

20 internal void Start()
21 {
22     if (!target) target = FindObjectOfType<Car>().transform;
23 }
24
25 |
26 [ExecuteInEditMode]
27 void LateUpdate()
28 {
29     if (target)
30     {
31         transform.position = new Vector3(
32             Mathf.SmoothDamp(transform.position.x, target.position.x, ref velocity, time),
33             Mathf.SmoothDamp(transform.position.y, target.position.y, ref velocity, time),
34             transform.position.z);
35     }
36 }

```

Skrypt „Car”

Służy do poruszania pojazdem. Każde koło ma własny napęd. Obracanie pojazdem jest możliwe tylko w powietrzu.

```
30 // Funkcja sprawdzająca czy oba koła znajdują się w powietrzu
31 public bool IsFlying
32 {
33     get
34     {
35         bool isFlying = true;
36         foreach (Wheel wheel in wheelScripts)
37             isFlying &= wheel.isFlying;
38         return isFlying;
39     }
40 }

43 protected internal void Start()
44 {
45     if (wheelJoints.Length == 0)
46         wheelJoints = GetComponentsInChildren<WheelJoint2D>(); // Zwraca typ komponentów obiektu lub jego pochodnych
47
48     if (wheelScripts.Length == 0)
49         wheelScripts = GetComponentsInChildren<Wheel>();
50
51     if (wheelJoints.Length > 0)
52     {
53         motor = wheelJoints[0].motor;
54         setWheelJoints(motor);
55     }
56 }

60 // Ustawia połączenia kół
61 private void setWheelJoints(JointMotor2D _motor)
62 {
63     foreach (WheelJoint2D w in wheelJoints)
64         w.motor = _motor;
65 }

68 // Funkcja, która pozwala na poruszanie się zarówno w powietrzu jak i w locie,
69 protected virtual void FixedUpdate()
70 {
71     if (!IsFlying)
72     {
73         motor.motorSpeed = -Input.GetAxis("Horizontal") * speed; // by strzałka w prawo odpowiadała za ruch do przodu
74         setWheelJoints(motor);
75     }
76     else
77     {
78         Vector3 rot = transform.eulerAngles;
79         rot.z = rot.z + Input.GetAxis("Horizontal") * Mathf.Sqrt(speed / 50);
80         transform.eulerAngles = rot;
81     }
82 }
```