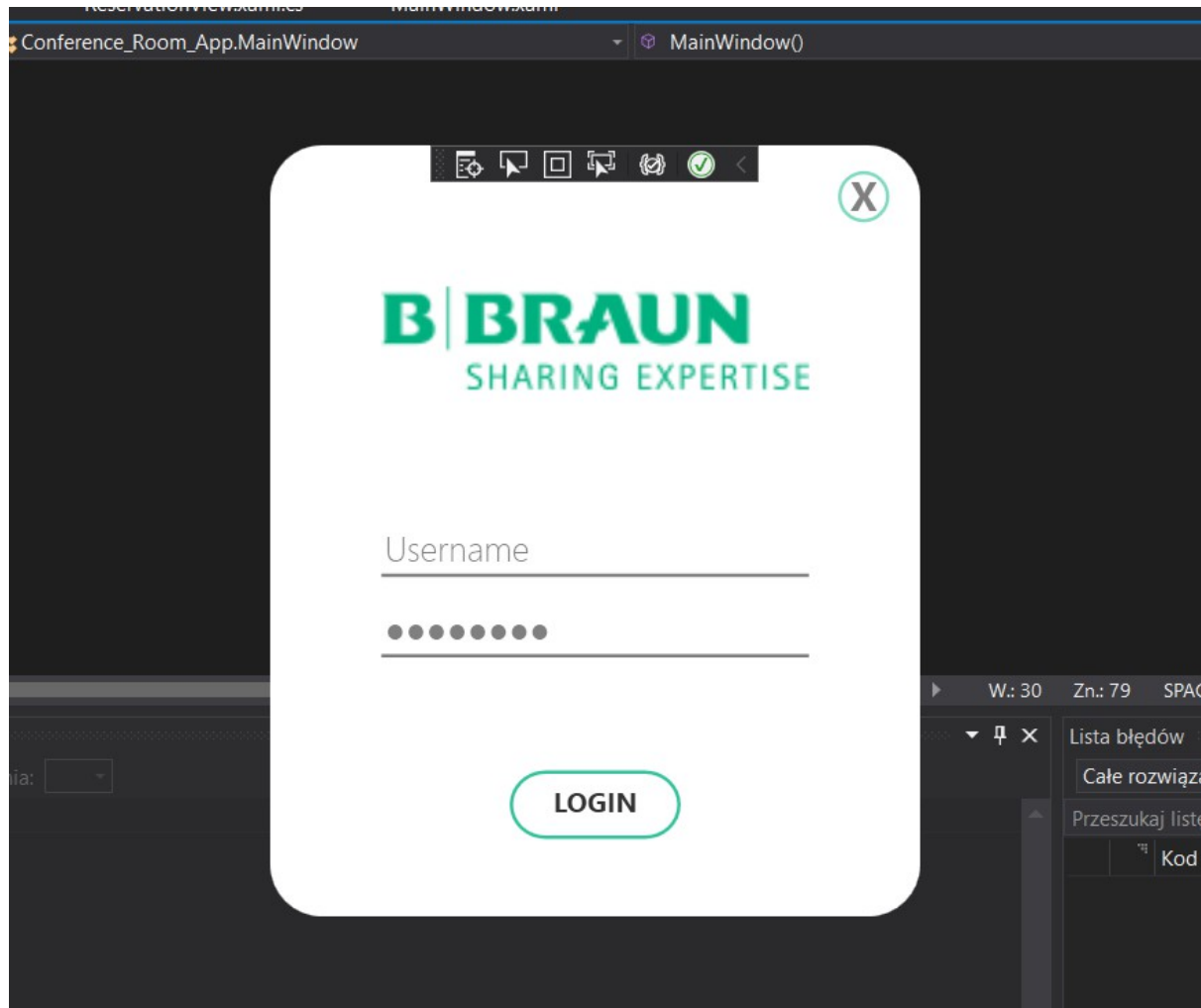


Paweł Wawrzyniak

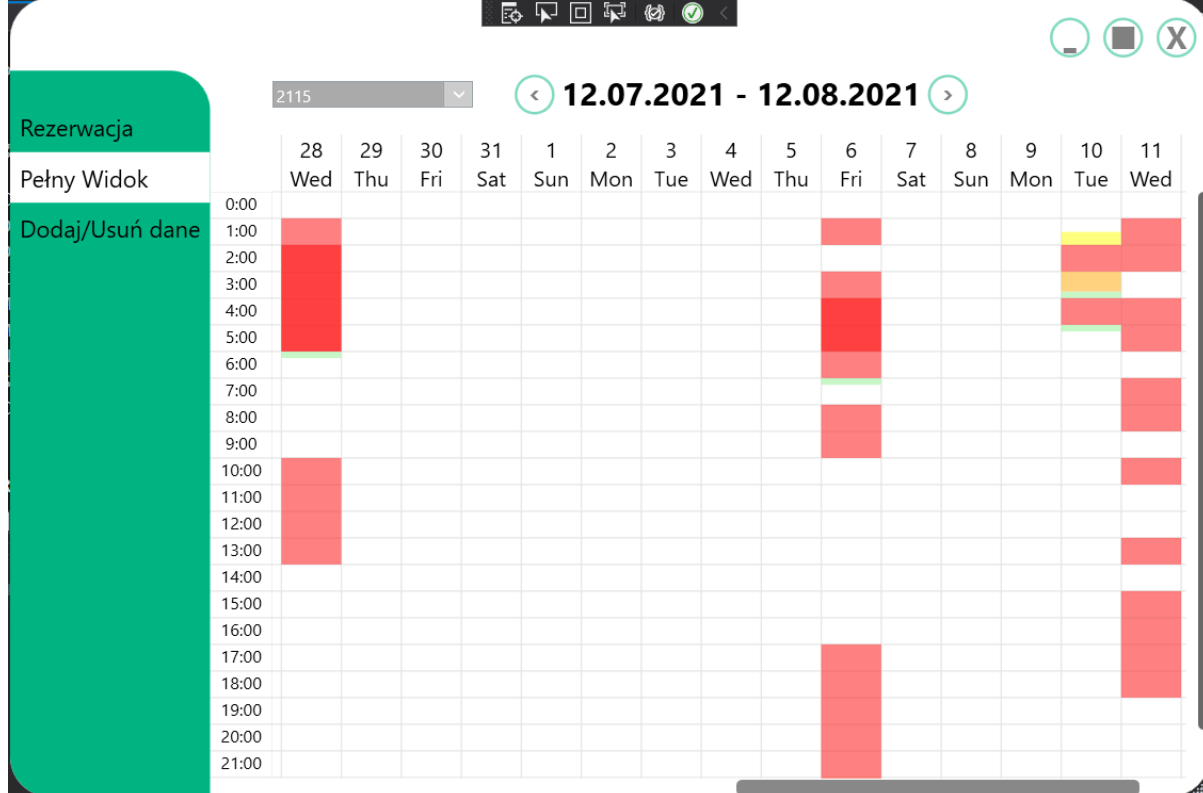
Portfolio - Programista

Zaprojektowanie i wykonanie pełnego programu

Rezerwacja Sal



Grid kalendarza :



```

public void createGrid()
{
    int daysQuantity = DateTime.DaysInMonth(chosenDate.Year, chosenDate.Month) + 1;
    int cellWidth = 45;
    int cellHeight = 28;
    int hours = 24;
    setDataRange();

    for (int i = 0; i < hours; i++)
    {
        RowDefinition hourRows = new RowDefinition()
        {
            Height = new GridLength(cellHeight)
        };
        RowDefinition rows = new RowDefinition()
        {
            Height = new GridLength(cellHeight)
        };
        myGrid.RowDefinitions.Add(hourRows);
        hourGrid.RowDefinitions.Add(hourRows);
    }

    for (int i = 0; i < daysQuantity; i++)
    {
        ColumnDefinition dataCols = new ColumnDefinition()
        {
            Width = new GridLength(cellWidth),
        };
        ColumnDefinition cols = new ColumnDefinition()
        {
            Width = new GridLength(cellWidth)
        };
        myGrid.ColumnDefinitions.Add(cols);
        dataGrid.ColumnDefinitions.Add(dataCols);
    }

    for (int i = 0; i < hours; i++)
    {
        TextBlock hourLbl = new TextBlock()
        {
            HorizontalAlignment = HorizontalAlignment.Center,
            VerticalAlignment = VerticalAlignment.Bottom,
            TextAlignment = TextAlignment.Center,
            FontSize = 13,
            Height = cellHeight,
            Width = cellWidth,
            Foreground = Brushes.Black,
            Text = i.ToString() + ":00"
        };
        Grid.SetRow(hourLbl, i);
        hourGrid.Children.Add(hourLbl);
    }

    for (int i = 0; i < daysQuantity; i++)
    {
        TextBlock dataLbl = new TextBlock()
        {
            FontSize = 16,
            Foreground = Brushes.Black,
            Text = chosenDate.AddDays(i).Day.ToString() + "\n" + chosenDate.AddDays(i).DayOfWeek.ToString().Remove(1),
            TextAlignment = TextAlignment.Center,
            TextWrapping = TextWrapping.Wrap
        };
        Grid.SetColumn(dataLbl, i);
        dataGrid.Children.Add(dataLbl);
    }
}

```

```

using (SqlConnection sqlConn = new SqlConnection(connectionString))
{
    try
    {
        refreshGrid(); //Wczytywanie danych

        if (refreshCalendar.SelectCalendar() != -1)
        {
            if (sqlConn.State == ConnectionState.Closed)
            {
                sqlConn.Open();
            }
            string query = $"select startYear, startMonth, startDay, startHour, startMinute, endYear, endMinute from Steps where RoomName = '{refreshCalendar.SelectCalendar()}'";
            using (SqlCommand command = new SqlCommand(query, sqlConn))
            {
                using (SqlDataReader reader = command.ExecuteReader())
                {
                    int[] selectedDates = new int[daysQuantity];
                    while (reader.Read())
                    {
                        //Wczytywanie danych z bazy
                        if (Convert.ToInt32(reader.GetInt32(0).ToString()) + "/" + reader.GetInt32(1).ToString() + "/" + reader.GetInt32(2).ToString() != chosenDate.ToString())
                        {
                            Convert.ToInt32(reader.GetInt32(0).ToString()) + "/" + reader.GetInt32(1).ToString() + "/" + reader.GetInt32(2).ToString() < chosenDate
                        }
                        for (int i = reader.GetInt32(0); i < reader.GetInt32(1); i++)
                        {
                            //Sprawdzanie czy podana data jest w przedziale daty ujętej w kalendarzu
                            TestBlock lbl = new TestBlock()
                            {
                                HorizontalAlignment = HorizontalAlignment.Center,
                                VerticalAlignment = VerticalAlignment.Center,
                                Background = Brushes.Black,
                                Width = calendarWidth,
                                Height = calendarHeight,
                                Opacity = 0.5,
                            };
                        }

                        if (reader.GetInt32(0) == i && reader.GetInt32(1) != 0)
                        {
                            if (reader.GetInt32(0) >= 0)
                            {
                                lbl.Width = calendarWidth / 4;
                                lbl.VerticalAlignment = VerticalAlignment.Bottom;
                                lbl.BackgroundColor = Brushes.LightGreen;
                            }
                            else if (reader.GetInt32(0) >= 10)
                            {
                                lbl.Width = calendarWidth / 2;
                                lbl.VerticalAlignment = VerticalAlignment.Bottom;
                                lbl.BackgroundColor = Brushes.Yellow;
                            }
                            else if (reader.GetInt32(0) >= 15)
                            {
                                lbl.Width = calendarWidth / 4 * 3;
                                lbl.VerticalAlignment = VerticalAlignment.Bottom;
                                lbl.BackgroundColor = Brushes.Orange;
                            }
                        }

                        if (reader.GetInt32(1) != chosenDate.Month)
                        {
                            Grid.SetColumn(lbl, reader.GetInt32(1) - chosenDate.Day + daysQuantity - 1);
                        }
                        else
                        {
                            Grid.SetColumn(lbl, reader.GetInt32(1) - chosenDate.Day);
                        }
                        Grid.SetRow(lbl, 1);
                        Grid.SetRowSpan(lbl, 1);
                    }
                }

                if (reader.GetInt32(0) != 0)
                {
                    TestBlock lbl = new TestBlock()
                    {
                        HorizontalAlignment = HorizontalAlignment.Center,
                        VerticalAlignment = VerticalAlignment.Top,
                        Background = Brushes.Black,
                        Opacity = 0.5,
                        Width = calendarWidth,
                    };

                    if (reader.GetInt32(0) < 15)
                    {
                        lbl.Width = calendarWidth / 4;
                        lbl.BackgroundColor = Brushes.LightGreen;
                    }
                    else if (reader.GetInt32(0) <= 10)
                    {
                        lbl.Width = calendarWidth / 2;
                        lbl.BackgroundColor = Brushes.Yellow;
                    }
                    else if (reader.GetInt32(0) <= 15)
                    {
                        lbl.Width = calendarWidth / 4 * 3;
                        lbl.BackgroundColor = Brushes.Orange;
                    }

                    Grid.SetRow(lbl, reader.GetInt32(1));
                    if (reader.GetInt32(1) != chosenDate.Month)
                    {
                        Grid.SetColumn(lbl, reader.GetInt32(1) - chosenDate.Day + daysQuantity - 1);
                    }
                    else
                    {
                        Grid.SetColumn(lbl, reader.GetInt32(1) - chosenDate.Day);
                    }
                    Grid.SetRowSpan(lbl, 1);
                }
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
    finally
    {
        sqlConn.Close();
    }
}
}

```

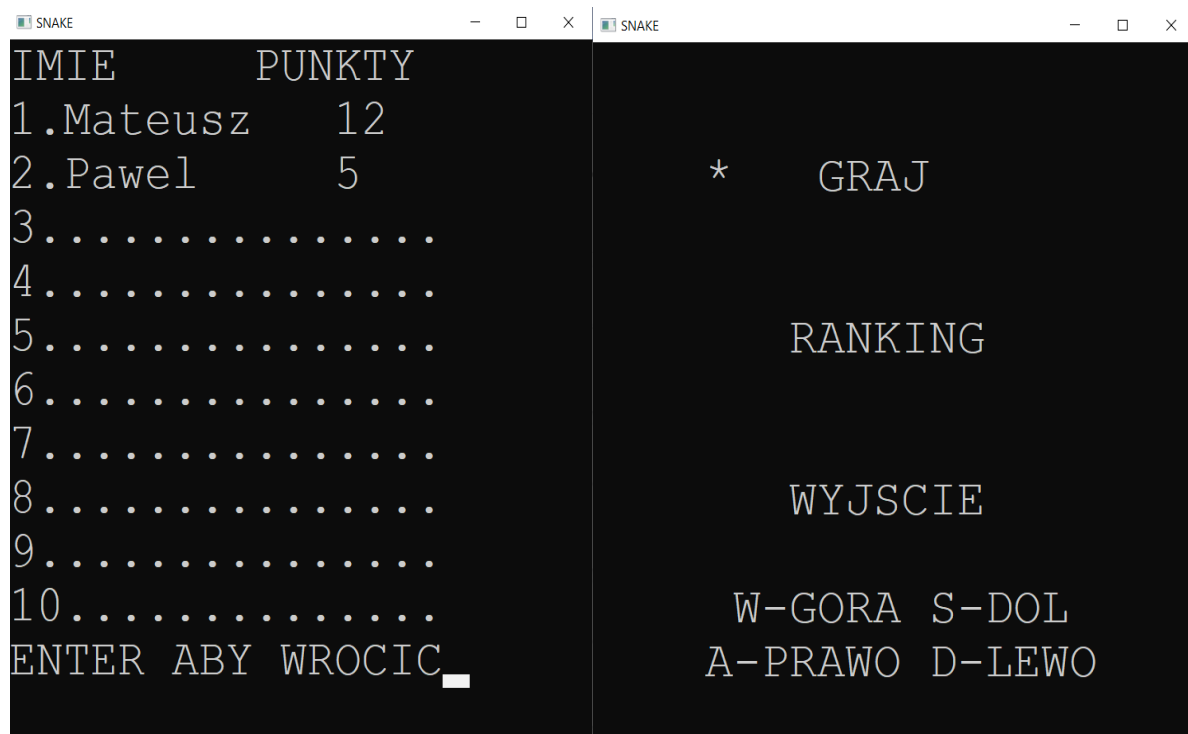
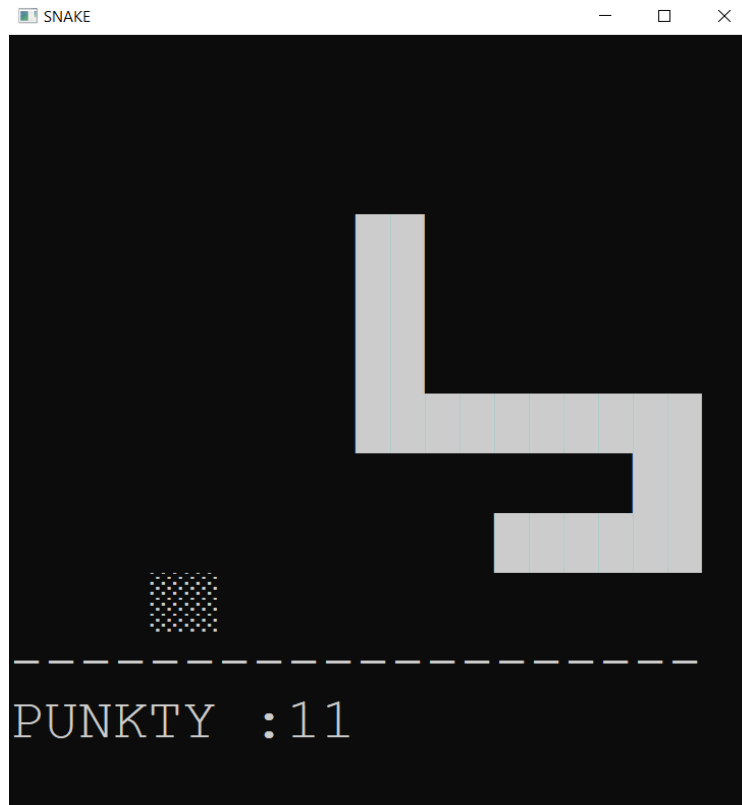
Najtrudniejszym zadaniem w tej aplikacji było wyświetlenie własnego gridu kalendarza.

Zaprojektowanie tej aplikacji wymagało ode mnie użycia całej mojej wiedzy na temat .Net oraz baz danych.

Snake 2020r.

Został wykonany w terminalu bez silnika - 272 linijki

github: <https://github.com/mrskaterr/Snake>



```

142 Snake* turnAndBuild(char turn, Snake* start) {
143     Snake* Help = new Snake;
144     if (turn == 'w' || turn == 'W') {
145         if (HeadX == 0) HeadX = Size - 1;
146         else HeadX--;
147     }
148     else if (turn == 's' || turn == 'S') {
149         if (HeadX == Size - 1) HeadX = 0;
150         else HeadX++;
151     }
152     else if (turn == 'd' || turn == 'D') {
153         if (HeadY == Size - 1) HeadY = 0;
154         else HeadY++;
155     }
156     else if (turn == 'a' || turn == 'A') {
157         if (HeadY == 0) HeadY = Size - 1;
158         else HeadY--;
159     }
160
161     if ((* (Space + HeadX) + HeadY) == Point) {
162         Score++;
163         (* (Space + HeadX) + HeadY) = Body;
164         Help->BodyPart = (* (Space + HeadX) + HeadY);
165         Help->Next = start;
166         return Help;
167     }
168     else if (Score == 0) {
169         Help->BodyPart = (* (Space + HeadX) + HeadY);
170         (* (Space + HeadX) + HeadY) = Body;
171         Help->Next = NULL;
172         (* (start).BodyPart) = Blank;
173         start = NULL;
174         return Help;
175     }
176     else if ((* (Space + HeadX) + HeadY) == Blank) {
177         Snake* help2 = new Snake;
178         Help->BodyPart = (* (Space + HeadX) + HeadY);
179         (* (Space + HeadX) + HeadY) = Body;
180         Help->Next = start;
181         help2 = start;
182         for (int i = 0; i < Score; i++) {
183             start = help2;
184             if (help2 == NULL) {
185                 SaveScore();
186                 Ranking();
187             }
188             help2 = help2->Next;
189             if (help2->Next == NULL) {
190                 (* (help2).BodyPart) = Blank;
191                 help2 = NULL;
192                 start->Next = NULL;
193                 return Help;
194             }
195         }
196     }
197     return NULL;

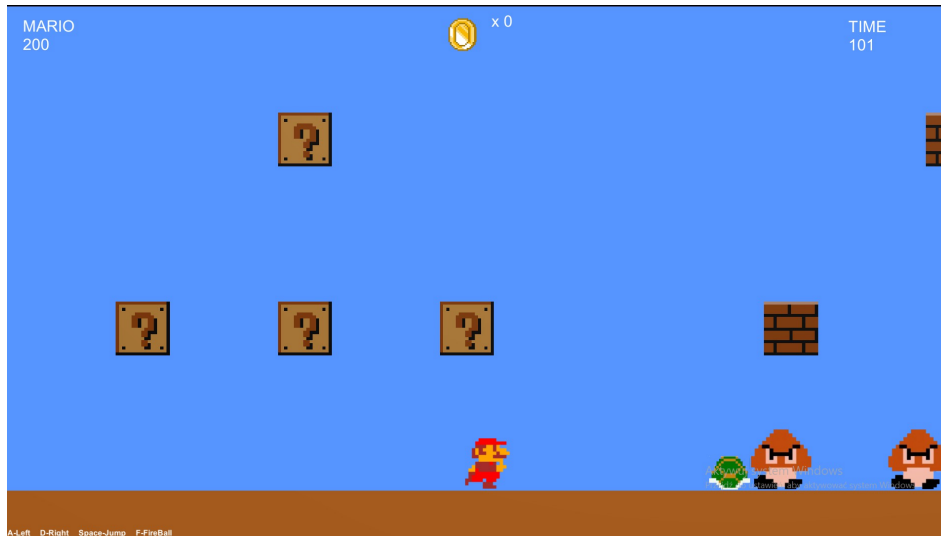
```

Nad tą częścią kodu przesiedziałem najwięcej czasu. Jest on odpowiedzialny za poruszanie się węża, wydłużanie się go oraz zdobywanie punktów.

MARIO 2020r.

Unity C#

github: <https://github.com/mrskaterr/Mario>



Pierwszy poziom Mario wykonany wyłącznie przeze mnie.

```
4
5
6 public class Turtle : MonoBehaviour
7 {
8     public float Distance;
9     public float Speed;
10    float PrivateSpeed = 0;
11    public GameObject Player;
12    public GameObject Mushroom;
13    public GameObject Canvas;
14
15    Rigidbody2D rb;
16    void Start()
17    {
18        rb = GetComponent<Rigidbody2D>();
19    }
20
21    void Update()
22    {
23        if (PrivateSpeed == 0 && (gameObject.transform.position.x - Player.transform.position.x) <= Distance) PrivateSpeed = Speed; //Run Aktivation
24        rb.velocity = new Vector3(PrivateSpeed, 0, 0);
25    }
26    private void OnCollisionEnter2D(Collision2D collision)
27    {
28
29        if (collision.gameObject == Player && collision.gameObject.GetComponent<Rigidbody2D>().velocity.y < 0) //Tutrlie transform to Turtle2
30        {
31            GameObject.FindGameObjectWithTag("Canvas").GetComponent<TimeScoreCoin>().AddScore(200);
32            gameObject.transform.GetChild(0).gameObject.SetActive(false);
33            gameObject.transform.GetChild(1).gameObject.SetActive(true);
34            if (collision.gameObject.transform.position.x > gameObject.transform.position.x) gameObject.GetComponent<Turtle>().PrivateSpeed = Mathf.Abs(PrivateSpeed)*1.5f; //RIGHT
35            if (collision.gameObject.transform.position.x < gameObject.transform.position.x) gameObject.GetComponent<Turtle>().PrivateSpeed = -Mathf.Abs(PrivateSpeed)*1.5f; //LEFT
36
37        }
38
39        else if (collision.gameObject == Player && collision.gameObject.transform.localScale.x > 1)
40        {
41            collision.gameObject.GetComponent<Movement>().Jump2();
42            collision.gameObject.transform.localScale = new Vector3(1, 1, 1);
43        }
44        else if (collision.gameObject == Player) //Player DIE
45        {
46            collision.gameObject.GetComponent<PlayerDead>().enabled = true;
47        }
48        else if (collision.gameObject == Mushroom && PrivateSpeed != Speed) //Mushrom Die
49        {
50            collision.gameObject.SetActive(false);
51            Canvas.GetComponent<TimeScoreCoin>().AddScore(200);
52        }
53        else if (rb.velocity.x == 0) PrivateSpeed *= -1; //Change
54    }
55 }
```

Najtrudniejszym zadaniem było stworzenie movementu dla żółwia. Żółw mógł Cię zaatakować lecz gdy dobrze się na niego skoczy, unieszkodliwia się go i można posługiwać się nim jak piłką do gry oraz atakować inne potwory.

