|  |
| --- |
| Ostatni we wszechświecie |

Przemysław Szymański

Maciej Pozorski

Jakub Wachowicz

17.05.2021

Opiekun

**dr inż. Tacjana Niksa-Rynkiewicz**

Spis treści

1. Wprowadzenie 3

1.1 Cel gry 3

1.2 Kiedy się wygrywa? 3

1.3 Kiedy się przegrywa? 3

1.4 Co utrudnia zwycięstwo? 3

1.5 Co pomaga wygrać? 3

1.6 Sterowanie 3

1.7 Jak uruchomić grę? 3

2. Schemat klas UML 4

3. Rozwiązanie / Kod źródłowy 4

3.1 Język 4

3.2 Narzędzia 4

3.3 Kod programu 4

3.3.1 Struktura plików i katalogów 4

3.3.2 Characters.cs 5

3.3.2 Events.cs 6

3.3.3 Intro.cs 7

4.3.4 Ships.cs 9

3.3.6 Menu.cs 14

3.3.7 Program.cs 20

4. Podsumowanie 22

4.1 Co było łatwe? 22

4.2 Co było trudne? 22

4.3 Jak można rozbudować grę? 22

4.4 Czy zadanie sprawiło nam przyjemność? 22

* Wprowadzenie

W naszej grze wcielamy się w rolę operatora robota-pilota statku kosmicznego na którego pokładzie znajdują się zahibernowani ludzi. Statek musi jak najszybciej znaleźć planetę, która nada się do skolonizowania przez ludzi. Robot-pilot jest w stanie wykonywać takie czynności jak pilotowanie, naprawa statku czy ładowanie baterii, jednak to my decydujemy którą czynność robot będzie wykonywać w danym momencie.

1.1 Cel gry

Celem gry jest jak najszybsze dotarcie do planety nadającej się do zamieszkania przez uśpioną załogę statku.

1.2 Kiedy się wygrywa?

Grę wygrywa się docierając do planety możliwej do skolonizowania.

1.3 Kiedy się przegrywa?

Grę można przegrać doprowadzając do sytuacji gdy dalsza podróż statkiem jest niemożliwa np. Kompletne rozładowanie baterii robota, bądź całkowite zniszczenie robota lub statku.

1.4 Co utrudnia zwycięstwo?

Podczas podróży mogą wydarzyć się różne eventy, które będą od nas wymagać naprawy statku. Jako gracz musimy umiejętnie rozdzielać czas na podróż oraz czas na ładowanie akumulatorów tak by nie doprowadzić do ich kompletnego rozładowania.

1.5 Co pomaga wygrać?

W grze pomaga umiejętne rozdysponowanie czasem oraz uważne zwracanie uwagi na poziom baterii robota jak i stan techniczny robota bądź statku. Często naszym sojusznikiem może być zwykłe szczęście, ponieważ eventy są losowe.

1.6 Sterowanie

W grze “poruszamy” się wybierając akcję z menu wpisując odpowiadający jej numer

1.7 Jak uruchomić grę?

Grę można uruchomić za pomocą środowiska programistycznego bądź uruchamiając aplikację “ostatni.exe” znajdującą się w folderze release.

* Schemat klas UML



* Rozwiązanie / Kod źródłowy

3.1 Język

Kod programu został napisany w języku C#. Do napisania, debugowania oraz edytowania kodu został użyty edytor Visual Studio.

3.2 Narzędzia

Dla łatwiejszej współpracy podczas tworzenia programu zostało użyte oprogramowanie wykorzystujące system kontroli wersji – GitHub.

3.3 Kod programu

3.3.1 Struktura plików i katalogów



3.3.2 Characters.cs

Plik zawiera klasę Characters, która opisuje główną postać oraz funkcję ładowania baterii bohatera.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;    namespace Ostatni  {  public class Characters  {  public string RobotModel { get; set; }  public int StartBattery { get; set; }  public int Battery { get; set; }  public int StartRobotCondition { get; set; }  public int RobotCondition { get; set; }  public int RobotFailureRate { get; set; }    public Characters(int option)  {  if(option == 1)  {  RobotModel = "MT07";  StartBattery = Battery = 15;  StartRobotCondition = RobotCondition = 100;  RobotFailureRate = 30;  }  else if(option == 2)  {  RobotModel = "GSXR";  StartBattery = Battery = 10;  StartRobotCondition = RobotCondition = 150;  RobotFailureRate = 40;  }  else if(option ==3)  {  RobotModel = "Mark 39";  StartBattery = Battery = 12;  StartRobotCondition = RobotCondition = 120;  RobotFailureRate = 20;  }  }  public void ChargeBatteries(string reason)  {    bool start = true;  if ( RobotCondition <= StartRobotCondition/2)  {  Console.WriteLine("Brak mocy! Nie możesz naładować baterii.");  Console.WriteLine(reason);  Console.ReadLine();  Console.Clear();  start = false;  }  while (start == true)  {      Console.Write("|==========================================|\n");  Console.Write("| Podaj czas ładowania baterii (Exit = 0): |\n");  Console.Write("|==========================================|\n");    try  {  int time = Convert.ToInt32(Console.ReadLine());    if (time == 0)  {  start = false;  }  else  {  Battery += time;  if (Battery > StartBattery)  {  Battery = StartBattery;  }  start = false;  }  }  catch (System.FormatException)  {  Console.Clear();  Console.WriteLine("Tak to się nie uda! Podaj ile godzin.");  Console.ReadKey();  Console.Clear();  }    }        }  }  } |

3.3.2 Events.cs

Plik zawiera definicję listy obiektów losowych wydarzeń.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;    namespace Ostatni  {  public class Events  {  public int ID { get; set; }  public int Damage { get; set; }  public string Reason { get; set; }    public Events(int iD, int damage, string reason)  {  ID = iD;  Damage = damage;  Reason = reason;  }  }  } |

3.3.3 Intro.cs

Znajduje się w nim animacja otwierania nowej gry wraz z otoczką fabularną.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading;  using System.Threading.Tasks;    namespace Ostatni  {  class Intro  {  public static void AtlasIntro(string robotModel, string shipModel)  {  for (int i = 0; i < 25; i++)  {  Console.Write("#");  Thread.Sleep(50);  }  Console.Clear();  for (int i = 0; i < 3; i++)  {  Console.Write("System Atlas booting..");  Thread.Sleep(500);  Console.Clear();  Console.Write("System Atlas booting ");  Thread.Sleep(500);  Console.Clear();  }  Console.WriteLine("Downloading ILM (Intergalactic Light Map)");  for (int i = 0; i <= 100; i++)  {  Console.Write($"\rProgress: {i}% ");  Thread.Sleep(30);  }  Console.Clear();    Thread.Sleep(1250);    Console.Write(" ############################# \n");  Console.Write(" ############# ############# \n");  Thread.Sleep(300);  Console.Write(" ########### # ########### \n");  Console.Write(" ########## ### ########## \n");  Thread.Sleep(300);  Console.Write(" ######### ##### ######### \n");  Console.Write(" ####### ###### \n");  Thread.Sleep(300);  Console.Write(" ####### ######### ####### \n");  Console.Write(" ###### ########### ###### \n");  Thread.Sleep(300);  Console.Write(" ##### ############# ##### \n");  Console.Write(" #### ############### #### \n");  Thread.Sleep(300);  Console.Write(" ## ############### ## \n");  Console.Write(" ############################# \n");  Thread.Sleep(2500);  Console.Clear();  Console.Write("Witaj w PKP Programie Kont"); Thread.Sleep(150); Console.Write("roli Pozagalaktycznej Atlas.\n"); Thread.Sleep(150);  Console.Write("Zostałeś wybrany jako operator "); Thread.Sleep(150); Console.Write("humanoidalnego półautomatycznego \n"); Thread.Sleep(150);  Console.Write("robota-pilota statku naszej korporacji."); Thread.Sleep(150); Console.Write(" Przez dramatycznie pogarszający się\n"); Thread.Sleep(150);  Console.Write("stan kondycji naszej planety, postanowiliśmy "); Thread.Sleep(150); Console.Write("jak najszybciej wysłać statek z zahibernowanymi osobami\n");  Console.Write("przygotowanymi do kolonizacji nowej planety."); Thread.Sleep(150); Console.Write(" Jest to jedyny taki statek kosmiczny, \n");  Console.Write("o napędzie zakrzywiającym grawitację. "); Thread.Sleep(150); Console.Write("Jest to nasza jedyna nadzieja.\n");  Console.Write("Dlatego teraz przedstawimy Tobie twoje obowiązki. "); Thread.Sleep(150); Console.Write("Robot jest wstanie sam wykonywać pracę, takie jak pilotowanie\n");  Console.Write("lub konserwacja statku, lecz to tym musisz wydawać mu polecenia. "); Console.Write("Uznaliśmy że taki zabieg zminimalizuje ryzyko niepowodzenia.\n");  Console.Write("Nie zawiedź nas!");  Console.WriteLine("");  Console.WriteLine("");  Console.WriteLine("Wciśnij dowolny przycisk aby kontynuować.");  Console.ReadKey(true);  Console.Clear();  Thread.Sleep(500);  Console.Write("STAN STATKU \n");  Console.Write("Załoga:"); Thread.Sleep(500); Console.Write(" 500 osób"); Thread.Sleep(500); Console.Write(" 1 Robot ATLAS"); Thread.Sleep(500); Console.Write(" Model: {0}\n", robotModel);  Console.Write("Stan hibernacji: "); Thread.Sleep(500); Console.Write("100% Stabilny\n");  Console.Write("Statek: {0}, ", shipModel); Thread.Sleep(500); Console.Write("lokalizacja: "); Thread.Sleep(500); Console.Write("1 milion lat świetlnych od Galaktyki Andromedy\n");      }  }  } |

4.3.4 Ships.cs

Plik zawiera definicję statków którymi porusza się główny bohater. Znajdują się tam również metody odpowiadające za podróż i jej animację, czytanie i losowanie zdarzeń podczas podróży oraz metoda odpowiedzialna za naprawę statku.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;    namespace Ostatni  {  public class Ships  {  private static List<Events> \_events;  public static Characters Robot;  public string ShipModel { get; set; }  public int StartShipCondition { get; set; }  public int ShipCondition { get; set; }  public int Speed { get; set; }  public int ShipFailureRate { get; set; }  public int Accident { get; set; }  public string Reason { get; set; }  public int Distance { get; set; }    public Ships(int option)  {  if (option == 1)  {  ShipModel = "YZF-R1M";  StartShipCondition = ShipCondition = 100;  Speed = 40;  ShipFailureRate = 20;  Accident = 0;  Reason = "";  Distance = 0;  }  else if (option == 2)  {  ShipModel = "V-Storm1050";  StartShipCondition = ShipCondition = 120;  Speed = 30;  ShipFailureRate = 30;  Accident = 0;  Reason = "";  Distance = 0;  }  else if (option == 3)  {  ShipModel = "Fireblade-SP";  StartShipCondition = ShipCondition = 160;  Speed = 20;  ShipFailureRate = 10;  Accident = 0;  Reason = "";  Distance = 0;  }  }  public int Travel(int battery)  {  if (ShipCondition < 50)  {  Console.WriteLine("Nie mozęsz dokonać lotu! Napraw statek i spróbuj ponownie");  Console.WriteLine(Reason);  Console.ReadLine();  Console.Clear();  return 0;  }    while (true)  {  Console.Clear();  Console.WriteLine("|================================|");  Console.WriteLine("| Podaj czas podróży (Exit = 0): |");  Console.WriteLine("|================================|");    try  {  int time = Convert.ToInt32(Console.ReadLine());    if (time >= battery)  {  Console.Clear();  Console.WriteLine("Nie masz wystarczająco energii! Wciśnij dowolny przycisk i spróbuj ponownie.");  Console.ReadKey();  Console.Clear();  }  else if (time == 0)  {  Console.Clear();  return 0;  }  else  {  int x = TravelLoop(time);  // time =  Console.Clear();  return x;  }  }  catch (System.FormatException)  {  Console.Clear();  Console.WriteLine("Tak to się nie uda! Podaj ile godzin. Wciśnij dowolny przycisk i spróbuj ponownie.");  Console.ReadKey();  Console.Clear();  }  }  }  public int TravelLoop(int time)  {  LoadEvents();  int sumTime = 0;  bool x = false;    if(Accident == 1)  {  Console.WriteLine("Nie możesz podróżować!");  Console.WriteLine("Awaria!");  Console.WriteLine(Reason);  Console.ReadKey();  Console.Clear();  return 0;  }    while (true)  {  if (x==true)  {  Draw();  }  x = true;  Draw();  if (Accident == 0 && sumTime != time)  {  Distance += 1000 \* Speed;  sumTime += 1;  }  else if(Accident == 1)  {  Console.WriteLine("Awaria!");  Console.WriteLine(Reason);  Console.ReadKey();  Console.Clear();  return sumTime;  }  else if(Accident == 0 && sumTime == time)  {  Console.WriteLine("Podróż zakończona pomyślnie!");  Console.ReadKey();  Console.Clear();  return sumTime;  }  }  }  public void LoadEvents()  {  int IDcount = 0;    \_events = new List<Events>();  \_events.Add(new Events(IDcount, 10, "Uszkodzona antena radia."));  IDcount++;  \_events.Add(new Events(IDcount, 20, "Zerwana antena sonaru."));  IDcount++;  \_events.Add(new Events(IDcount, 30, "Uszkodzone poszycie na dziobie statku."));  IDcount++;  \_events.Add(new Events(IDcount, 40, "Brak sygnału z radaru. "));  IDcount++;  \_events.Add(new Events(IDcount, 50, "Uwaga! Utrata mocy! Generator uszkodzony, silniki są wyłączone."));  IDcount++;  \_events.Add(new Events(IDcount, 60, "Uwaga! Silnik uległ awarii! Uruchomienie trybu awaryjnego!"));  IDcount++;  \_events.Add(new Events(IDcount, 30, "Naruszenie szkieletu kadłuba statku."));  IDcount++;  \_events.Add(new Events(IDcount, 20, "Uszkodzenie poszycia w sródokręciu statku."));  IDcount++;  }  public void Draw()  {  Random random = new Random();  int draw = random.Next(1, 21);    if (draw <= \_events.Count)  {  foreach (Events events in \_events)  {  if(events.ID == draw - 1)  {  ShipCondition -= events.Damage;  Reason = events.Reason;    }  }  Accident = 1;  }  }  public int RepairShip(int startBattery, int battery)  {  bool start = true;  if (ShipCondition == StartShipCondition)  {  Console.WriteLine("Nie ma nic do naprawy.");  Console.ReadLine();  Console.Clear();  return 0;  }    while (start == true)  {  Console.Write("Bateria: ");  for (int i = 0; i < battery; i++)  {  Console.ForegroundColor = ConsoleColor.Green;  Console.Write("#");  }  for (int i = 0; i < startBattery - battery; i++)  {  Console.ForegroundColor = ConsoleColor.Gray;  Console.Write("#");  }  Console.ResetColor(); Console.WriteLine("");  Console.Write("Info:"); Console.Write(Reason + "\n");  Console.WriteLine("Stan statku: {0}",ShipCondition);  Console.WriteLine("|===========================================================|");  Console.WriteLine("| Podaj ile czasu chcesz przeznaczyć na naprawę: (Exit = 0) |");  Console.WriteLine("|===========================================================|");  try  {    int choise = int.Parse(Console.ReadLine());  if (choise > battery || choise > (StartShipCondition - ShipCondition))  {  Console.Clear();  Console.WriteLine("Zastanów się co chcesz zrobić. Spróbuj jeszcze raz.");  Console.ReadLine();  Console.Clear();  }  else if (choise == 0)  {  Console.Clear();  return 0;  }  else  {  Console.Clear();  Console.WriteLine("|=========================================================|");  Console.WriteLine("| Naprawa w toku.. Naciśnij dowolny klawisz aby zakończyć |");  Console.WriteLine("|=========================================================|");  Console.ReadLine();  ShipCondition += choise \* 10;  start = false;  Console.Clear();  Accident = 0;  return choise;  }  }  catch (System.FormatException)  {  Console.Clear();  Console.WriteLine("Tak to się nie uda! Podaj ile godzin. Wciśnij dowolny przycisk i spróbuj ponownie.");  Console.ReadKey();  Console.Clear();  }  }  return 0;  }  }  } |
|  |

3.3.6 Menu.cs

Główne menu naszej gry w którym spędzimy najwięcej czasu. To za pomocą niego wybieramy akcje które będą dziać się w naszej grze. Zawiera również metody odpowiedzialne za zapisywanie i wczytywanie stanu gry oraz sprawdzanie stanu gry (wygrana/przegrana).

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.IO;  using System.Threading;    namespace Ostatni  {  class Menu  {  public static Characters Robot;  public static Ships Ship;  public static bool intro;  public static int option;  public static bool start;  public static void MainMenu()  {  start = true;    if (intro == true)  {  bool opt = true;  while (opt == true)  {  Console.WriteLine("Wybierz model robota który ma odbyć misję:");  Console.WriteLine("|==========================================================|");  Console.WriteLine("|Nr| Model | Pojemność baterii | Wytrzmałość | Awaryjność |");  Console.WriteLine("|1.| MT07 | 15 | 100 | 30% |");  Console.WriteLine("|2.| GSXR | 10 | 150 | 40% |");  Console.WriteLine("|3.| CBR | 12 | 120 | 20% |");  Console.WriteLine("|==========================================================|");  Console.Write("Twój wybór: ");    try  {  option = int.Parse(Console.ReadLine());  if (option == 1 || option == 2 || option == 3)  {  Robot = new Characters(option);  opt = false;  Console.Clear();  }  else  {  Console.Clear();  Console.WriteLine("Nie ma takiej opcji! Spróbuj ponownie.");  Console.ReadKey();  Console.Clear();  }  }  catch (System.FormatException)  {  Console.Clear();  Console.WriteLine("Nieznane polecenie! Wciśnij dowolny przycisk i spróbuj ponownie.");  Console.ReadKey();  Console.Clear();  }  }    while (opt == false)  {  Console.WriteLine("Wybierz model statku który ma odbyć misję:");  Console.WriteLine("|=======================================================|");  Console.WriteLine("|Nr| Model | Wytrzmałość | Szybkość | Awaryjność |");  Console.WriteLine("|1.| YZF-R1M | 100 | 40 | 20% |");  Console.WriteLine("|2.| V-Storm1050 | 120 | 30 | 30% |");  Console.WriteLine("|3.| Fireblade-SP | 160 | 20 | 10% |");  Console.WriteLine("|=======================================================|");  Console.Write("Twój wybór: ");    try  {  option = int.Parse(Console.ReadLine());  if (option == 1 || option == 2 || option == 3)  {  Ship = new Ships(option);  opt = true;  Console.Clear();  }  else  {  Console.Clear();  Console.WriteLine("Nie ma takiej opcji! Spróbuj ponownie.");  Console.ReadKey();  Console.Clear();  }  }  catch (System.FormatException)  {  Console.Clear();  Console.WriteLine("Nieznane polecenie! Wciśnij dowolny przycisk i spróbuj ponownie.");  Console.ReadKey();  Console.Clear();  }  }    Intro.AtlasIntro(Robot.RobotModel, Ship.ShipModel);  Console.WriteLine("Wciśnij dowolny przycisk, aby kontynuować.");  Console.ReadKey();  Console.Clear();  intro = false;  }    while (start == true)  {  Console.Write("Bateria:");  Thread.Sleep(25);  for (int i = 0; i < Robot.Battery; i++)  {  Console.ForegroundColor = ConsoleColor.Green;  Console.Write("#");  Thread.Sleep(25);  }  for (int i = 0; i < Robot.StartBattery - Robot.Battery; i++)  {  Console.ForegroundColor = ConsoleColor.Gray;  Console.Write("#");  Thread.Sleep(25);  }  Console.WriteLine("");  Console.ResetColor();  Console.WriteLine("Status statku: {0} Przebyte km: {1}", Ship.ShipCondition, Ship.Distance);    Console.WriteLine("|==============================|");  Console.WriteLine("| 1. Podróż |");  Console.WriteLine("| 2. Ładowanie baterii |");  Console.WriteLine("| 3. Naprawa |");  Console.WriteLine("| 4. Zapisz |");  Console.WriteLine("| 0. Wyjdź |");  Console.WriteLine("|==============================|");    Console.WriteLine("Wybierz opcję: ");    try  {  int menuOption = int.Parse(Console.ReadLine());    switch (menuOption)  {  case 1:  Console.Clear();  int startShipCondition = Ship.ShipCondition;  Robot.Battery -= Ship.Travel(Robot.Battery);  startShipCondition -= Ship.ShipCondition;  Robot.RobotCondition -= startShipCondition;  Winner();  Loser();  break;  case 2:  Console.Clear();  Robot.ChargeBatteries(Ship.Reason);  //Winner();  //Loser();  Console.Clear();  break;  case 3:  Console.Clear();  Robot.Battery -= Ship.RepairShip(Robot.StartBattery, Robot.Battery);  Winner();  Loser();  Console.Clear();  break;  case 4:  Console.Clear();  Save();  break;  case 0:  start = false;  Console.Clear();  break;  default:  Console.Clear();  Console.WriteLine("Nie ma takiej opcji! Wciśnij dowolny przycisk i spróbuj ponownie.");  Console.ReadKey();  Console.Clear();  break;  }  }  catch (System.FormatException)  {  Console.Clear();  Console.WriteLine("Nieznane polecenie! Wciśnij dowolny przycisk i spróbuj ponownie.");  Console.ReadKey();  Console.Clear();  }  }  }  public static void Save()  {  string path = @"stats.txt";  using (StreamWriter sw = File.CreateText(path))  {  sw.WriteLine(Robot.RobotModel);  sw.WriteLine(Robot.StartBattery);  sw.WriteLine(Robot.Battery);  sw.WriteLine(Robot.StartRobotCondition);  sw.WriteLine(Robot.RobotCondition);  sw.WriteLine(Robot.RobotFailureRate);    sw.WriteLine(Ship.ShipModel);  sw.WriteLine(Ship.StartShipCondition);  sw.WriteLine(Ship.ShipCondition);  sw.WriteLine(Ship.Speed);  sw.WriteLine(Ship.ShipFailureRate);  sw.WriteLine(Ship.Accident);  sw.WriteLine(Ship.Reason);  sw.WriteLine(Ship.Distance);  }    Console.WriteLine("Pomyślnie zapisano naciśnij dowolny klawisz aby kontynuować");  Console.ReadKey();  Console.Clear();  }  public static void Load()  {  Robot = new Characters(1);  Ship = new Ships(1);    string path = "stats.txt";  if (File.Exists(path))  {  string[] read = File.ReadAllLines(path);    Robot.RobotModel = (string)read[0];  Robot.StartBattery = Convert.ToInt32(read[1]);  Robot.Battery = Convert.ToInt32(read[2]);  Robot.StartRobotCondition = Convert.ToInt32(read[3]);  Robot.RobotCondition = Convert.ToInt32(read[4]);  Robot.RobotFailureRate = Convert.ToInt32(read[5]);    Ship.ShipModel = (string)read[6];  Ship.StartShipCondition = Convert.ToInt32(read[7]);  Ship.ShipCondition = Convert.ToInt32(read[8]);  Ship.Speed = Convert.ToInt32(read[9]);  Ship.ShipFailureRate = Convert.ToInt32(read[10]);  Ship.Accident = Convert.ToInt32(read[11]);  Ship.Reason = (string)read[12];  Ship.Distance = Convert.ToInt32(read[13]);    intro = false;  MainMenu();  }  else  {  Console.WriteLine("Nie posiadasz zapisu gry!");  Console.ReadLine();  Console.Clear();  }  }  public static void Winner()  {  if(Ship.Distance >= 1000000)  {  Console.Clear();  Console.WriteLine("Wygrałeś");  Console.ReadKey();  Console.Clear();  start = false;  }  }  public static void Loser()  {  if(Robot.Battery <= 0 || Robot.RobotCondition <= 0 || Ship.ShipCondition <= 0)  {  Console.Clear();  Console.WriteLine("Przegrałeś");  Console.WriteLine(Robot.Battery);  Console.WriteLine(Robot.RobotCondition);  Console.WriteLine(Ship.ShipCondition <= 0);  Console.ReadKey();  Console.Clear();  start = false;  }  }  }  } |

3.3.7 Program.cs

Plik zawiera podstawowe menu z informacjami oraz możliwością wczytania zapisu gry lub utworzenie nowej gry.

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;    namespace Ostatni  {  class Program  {  static void Main(string[] args)  {  bool start = true;    while (start == true)  {  Console.WriteLine("|==============================|");  Console.WriteLine("| 1. Nowa gra |");  Console.WriteLine("| 2. Wczytaj grę |");  Console.WriteLine("| 3. Zasady i informacje |");  Console.WriteLine("| 4. Autorzy |");  Console.WriteLine("| 0. Wyjdź |");  Console.WriteLine("|==============================|");    Console.WriteLine("Wybierz opcję: ");    try  {  int option = int.Parse(Console.ReadLine());    switch (option)  {  case 1:  Console.Clear();  Menu.intro = true;  Menu.MainMenu();  Console.Clear();  break;  case 2:  Console.Clear();  Menu.Load();  break;  case 3:  Console.Clear();  Console.WriteLine("|===========================================================|");  Console.WriteLine("| Twoim zadaniem jest przemierzanie bezkresu kosmosu |");  Console.WriteLine("| w poszukiwaniu nowej planety dla życia ludzkiego. |");  Console.WriteLine("| Uważaj na awarie, podejmuj taktyczne decyzje to w twoich |");  Console.WriteLine("| rękach jest przyszłość gatunku ludzkiego. |");  Console.WriteLine("|===========================================================|");  Console.WriteLine("Wciśnij dowolny przycisk, aby wrócić do menu.");  Console.ReadKey();  Console.Clear();  break;  case 4:  Console.Clear();  Console.WriteLine("|==============================|");  Console.WriteLine("| Maciej Pozorski |");  Console.WriteLine("| Przemysław Szymański |");  Console.WriteLine("| Jakub Wachowicz |");  Console.WriteLine("|==============================|");  Console.WriteLine("Wciśnij dowolny przycisk, aby wrócić do menu.");  Console.ReadKey();  Console.Clear();  break;  case 0:  start = false;  break;  default:  Console.Clear();  Console.WriteLine("Nie ma takiej opcji! Wciśnij dowolny przycisk i spróbuj ponownie.");  Console.ReadKey();  Console.Clear();  break;  }  }  catch (System.FormatException)  {  Console.Clear();  Console.WriteLine("Nieznane polecenie! Wciśnij dowolny przycisk i spróbuj ponownie.");  Console.ReadKey();  Console.Clear();  }  }  }  }  } |

* Podsumowanie

4.1 Co było łatwe?

Stworzenie wizualnej struktury programu.

4.2 Co było trudne?

Swobodny przepływ zmiennych między klasami.

4.3 Jak można rozbudować grę?

* Dodanie większej ilości eventów
* Dodanie losowej awaryjności statku
* Dodanie losowej awaryjności robota
* Wczytywanie eventów z pliku

4.4 Czy zadanie sprawiło nam przyjemność?

Choć proces tworzenia przysporzył nam trudności, wszyscy czegoś się nauczyliśmy przez co czujemy satysfakcję z projektu.