VOIDCRAFT THE GAME

SURVIVAL RPG

Dokumentacja

Skład

Piotr Zuber
Jan Snopek
Alan Biegun
Dawid Ryl
Konrad Sladkowski

1) Wprowadzenie

a) Cel projektu

Projekt „VoidCraft The Game” ma na celu stworzenie gry na urządzenia mobilne działające pod systemem Android. Jednym z głównych celów tego projektu jest stworzenie aplikacji łatwej w rozbudowie tak, aby w przyszłości projekt mógł być nadal rozwijany.

Gracz będzie doświadczał rozgrywki w perspektywie 2D z widokiem z góry na swoją postać. Całość gry przedstawiona została w konwencji survival-rpg, gdzie gracz dowolnie eksplorując udostępniony mu świat, może wchodzić z nim w interakcje w celu stworzenia narzędzi niezbędnych do przeżycia postaci.

Struktura oraz mechanizmy gry zostać miały stworzone przez nas od podstaw, aby poznać lepiej zasady projektowania tego typu aplikacji. Jednymi z najważniejszych celów jaki sobie postawiliśmy, były nauka poprawnego rozplanowywania prac, poznanie nowych narzędzi służących do tworzenia takich aplikacji jak również poprawienie efektywności naszej pracy grupowej.

b) Wymagania funkcjonalne

- Możliwość interakcji ze środowiskiem poprzez zbieranie przedmiotów
- Tworzenie przedmiotów przez system craftingu
- Tworzenie budynków
- Otwarty świat
- System dnia i nocy
- Możliwość tworzenia własnych map za pomocą osobnej aplikacji

2) Dokumentacja użytkownika

a) O programie

VoidCraft The Game to gra, która postawi Cię przed wyzwaniem, jakim jest przeżycie na tajemniczej wyspie. Każda noc może okazać się Twoją ostatnią. Czy dasz radę przetrwać?

W Twoje ręce trafiła właśnie gra, która daje Ci możliwość eksploracji otwartego świata oraz poznawanie wciągającej historii bezimiennej kobiety. Specjalnie dla Ciebie opracowaliśmy system carftingu, który pozwoli ci tworzyć narzędzia służące do przetrwania oraz samemu zmieniać swoje otoczenie poprzez stawianie budynków.

Dzięki naszej grze przeżyjesz wspaniałą przygodę oraz poznasz historię, która **zmrozi Ci nie raz krew w żyłach**.

Czy przeżyjesz?

b) Wymagania techniczne

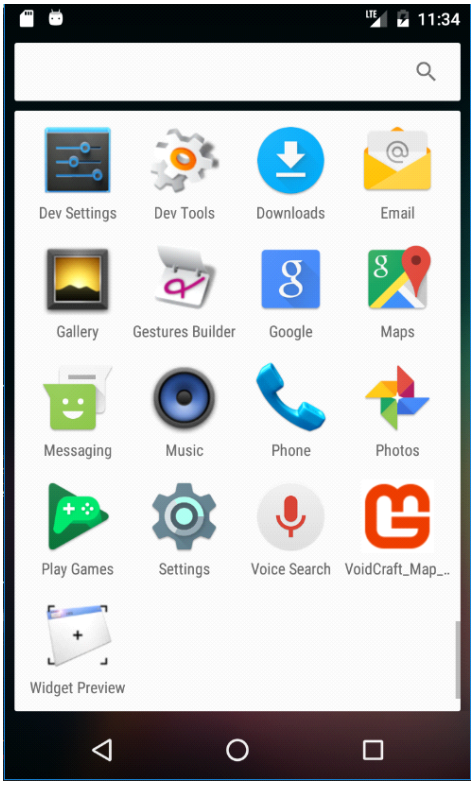
- Telefon z systemem Android w wersji 4.4 lub wyższej
- 512 MB pamięci ram lub więcej
- Przynajmniej 100 MB wolnego na dysku
- Należy przyznać aplikacji prawa do modyfikacji danych na karcie SD

c) Instalacja

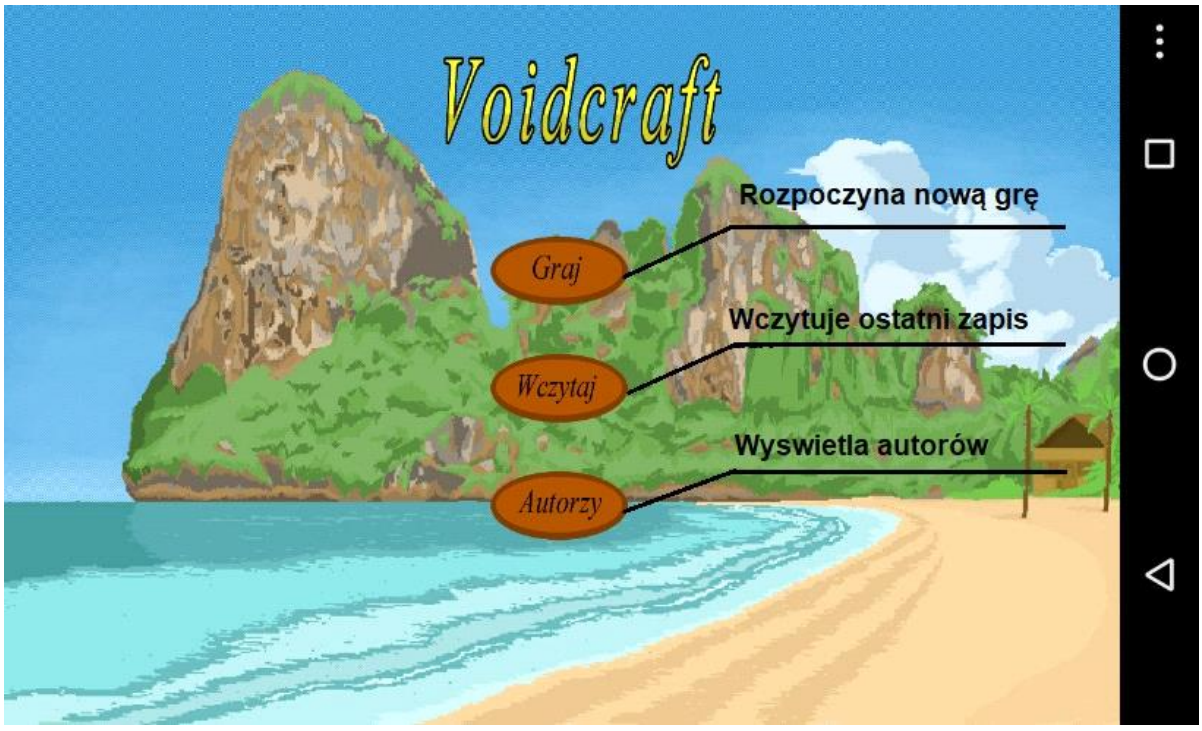
Należy uruchomić plik z rozszerzeniem *.apk oraz postępować według instrukcji. Jeśli aplikacja została pobrana z innego źródła niż SklepPlay to należy wyłączyć w ustawieniach telefonu opcję „Instaluj tylko z zaufanych źródeł”.

d) Korzystanie z programu

Aplikację należy uruchamiać z poziomu menu głównego telefonu. Należy ja znaleźć w menu lub na ekranie głównym urządzenia a następnie dotknąć jej ikony.



W wyniku powyższych czynności, naszym oczom ukaże się menu główne gry



Po naciśnięciu przycisku **Graj** ukaże się nam główny widok gry

W lewym dolnym rogu znajdują się strzałki odpowiadające za **poruszanie się postaci po mapie**. Kierunki strzałek odpowiadają kierunkom w których porusza się postać po przytrzymaniu danego klawisza. W prawym dolnym rogu znajdują kolejne dwa przyciski.

Przycisk A odpowiada za **interakcję z otoczeniem**, natomiast **przycisk B otwiera dziennik z zadaniami**, gdzie znajdują się wpisy informujące o postępach zadań.

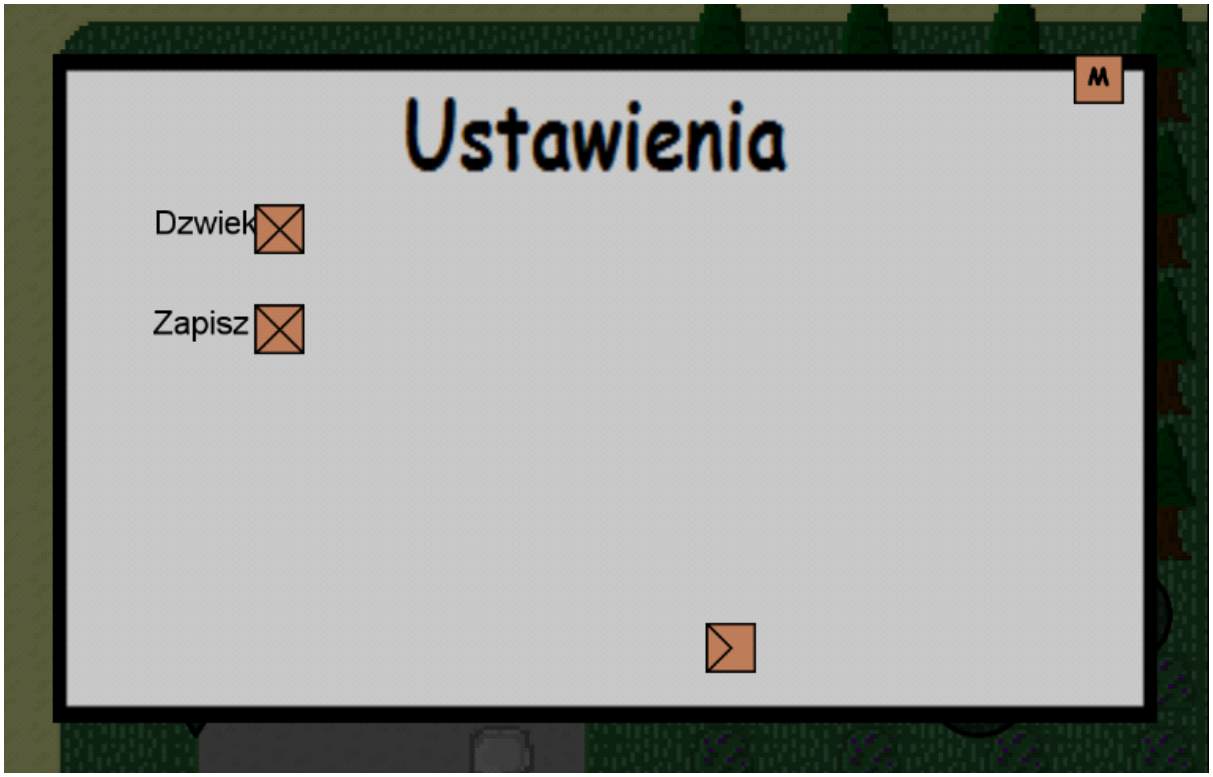
Poza menu głównym, w grze znajduje się także przycisk umożliwiający przejście do paneli, które umożliwiają podjęcie czynności opisanych poniżej.

Aby włączyć menu w grze należy przycisnąć klawisz z literą M znajdujący się w prawym górnym rogu.

Menu w grze umożliwia graczowi:

- Zapis gry
- Włączenie / wyłączenie dźwięku
- Przegląd ekwipunku gracza
- Tworzenie przedmiotów
- Przeglądanie dziennika zadań

Po włączeniu menu w grze pojawi się następujące okno:



Jest to okno ustawień. Po kliknięciu przycisku znajdującego się w prawym górnym rogu **nastąpi przeniesienie gracza z powrotem do gry**.

Aby **wyłączyć / włączyć dźwięk** należy kliknąć brązowy przycisk z krzykiem po lewej stronie podpisany jako Dźwięk.

Przycisk podpisany jako **Zapisz** używany jest do **zapiania postępu gry**. Od tego momentu gra może być wczytana z poziomu głównego menu.

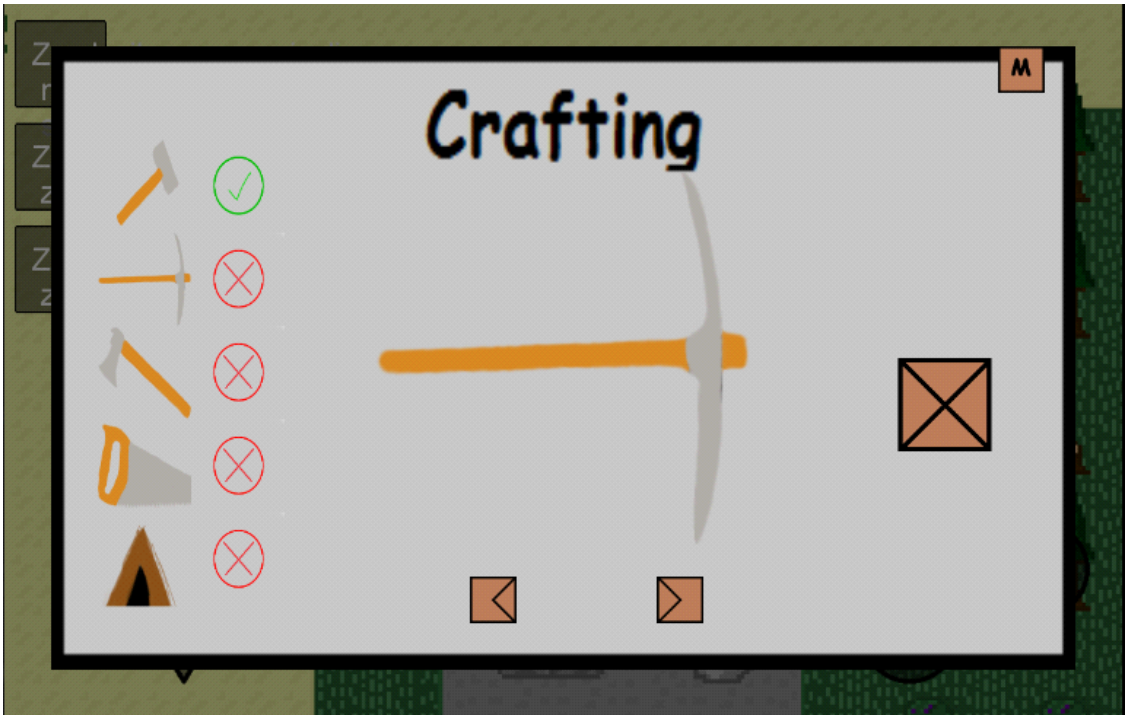
Przyciski znajdujące się na dole odpowiadają za **nawigację po menu**. W przypadku okna ustawień kliknięcie przycisku nawigującego przeniesie gracza do okna ekwipunku.

Po lewej stronie okna widzimy wszystkie **surowce jakie aktualnie posiada gracz**.

W menu ekwipunku gracz może korzystać z zebranych przedmiotów. Podczas gry poziom naszych statystyk będzie cały czas malał. Kliknięcie na ikonę **jedzenia odnowi poziom głodu oraz zdrowia**, natomiast Ikona wody **odnowi poziom pragnienia.**

Po prawej stronie okna widzimy narzędzia które gracz może stworzyć. Gdy gracz zdobędzie dane narzędzie jego kolor w ekwipunku zmieni się. Kolor czarny oznacza że gracz nie posiada danego narzędzia.

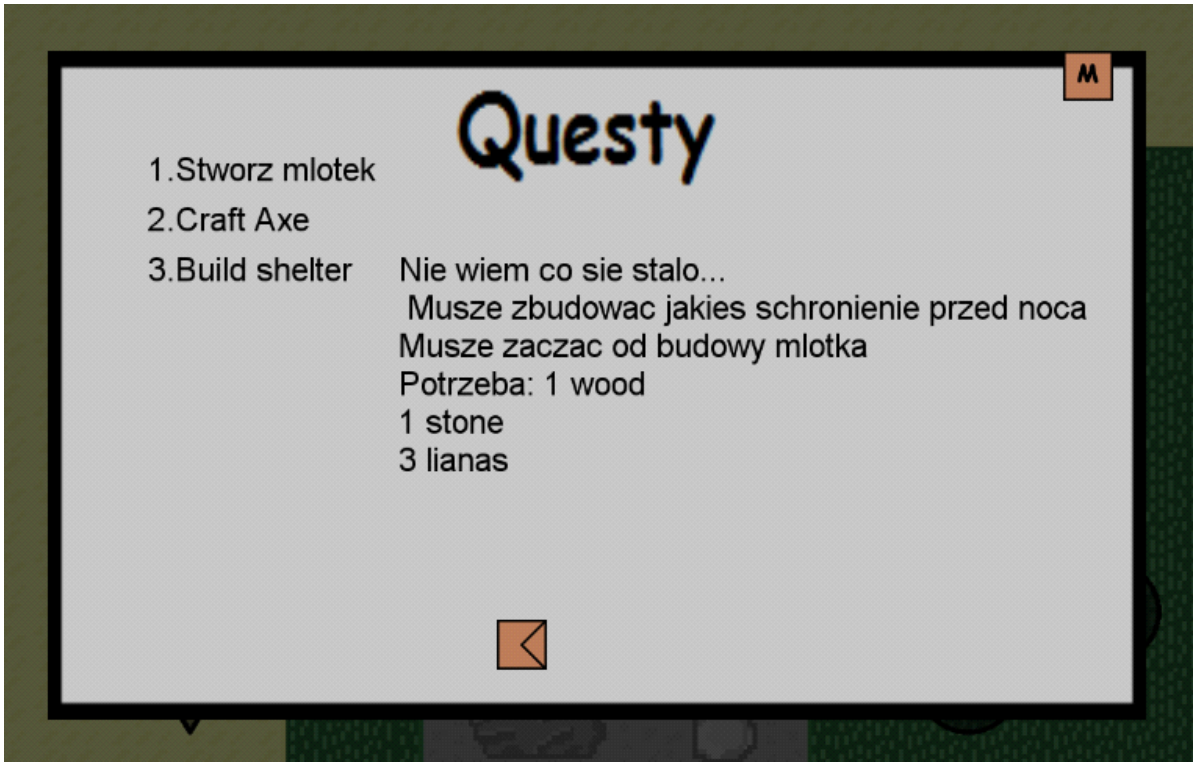
Przycisk nawigacyjny znajdujący się po lewej **przeniesie gracza do menu ustawień**, natomiast przycisk po prawej **otworzy menu tworzenia przedmiotów**.



Powyżej przedstawiony został wygląd interfejsu do tworzenia przedmiotów. Po lewej znajdują się ikony przedmiotów które gracz może stworzyć. Przy każdym przedmiocie znajduje się **znacznik informujący gracza czy dany przedmiot jest przez niego posiadany**.

Kliknięcie w wybrany przedmiot oznacza **wybranie go jako aktualny przedmiot do utworzenia**. Pojawi się on na środku okna, i jeżeli gracz posiada wystarczającą ilość surowców można go stworzyć. Aby **zatwierdzić nasz wybór i stworzyć przedmiot** należy kliknąć przycisk znajdujący się po prawej stronie okna.

Przycisk nawigacyjny po lewej **przeniesie gracza do ekwipunku**, a przycisk po prawej **do dziennika zadań.**



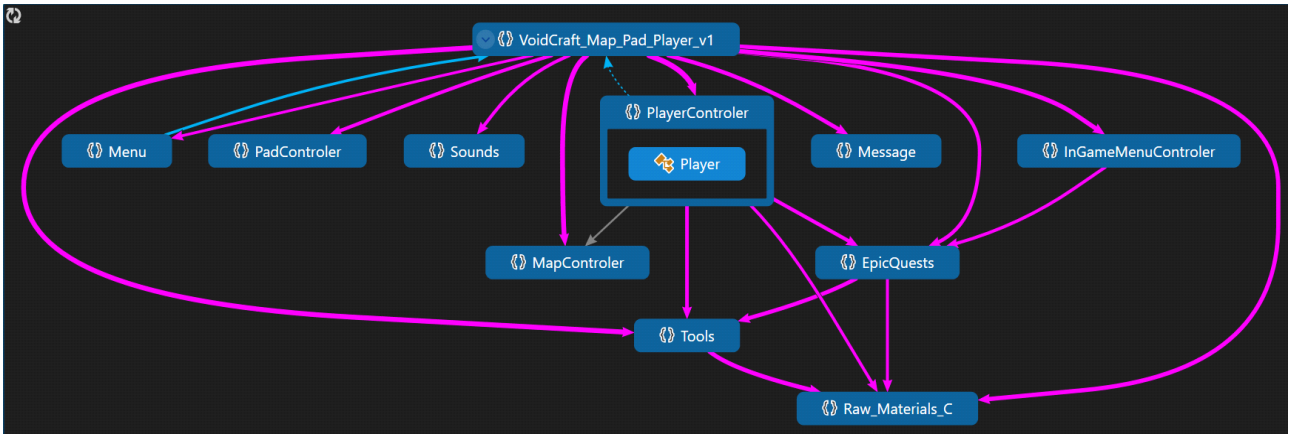
W oknie tym wyświetlane są informacje o zadaniach wykonywanych przez gracza. Po lewej stronie okna **znajdują się nazwy zadań**, natomiast na środku **wyświetlony jest aktualny wpis** zawierający wskazówki dla gracza.

Przycisk nawigacyjny na dole **przenosi gracza do menu tworzenia przedmiotów**.

3) Dokumentacja techniczna

a) Podział programu

Na poniższym rysunku jest przedstawiony graf, który odwzorowuje połączenia między głównymi klasami programu.



b) Najważniejsze klasy

Klasa Player

Klasa ta jest odpowiedzialna za przechowywanie wszystkich statystyk gracza takich jak zdrowie, głód, posiadane przedmioty ale także zmienne odpowiadające za poprawną animację postaci np. ilość klatek w szablonie postaci.

Odpowiada ona także za rysowanie animacji w głównej pętli programu, ponadto jest także serializowana do pliku XML, z którego jest odczytywany stan zapisanej gry.

Serializacja tej klasy odbywa się za pomocą metod LoadPlayer oraz SavePlayer.

```
public static Player LoadPlayer()
    {
      var store = IsolatedStorageFile.GetUserStoreForApplication();
      XmlSerializer xmlFormat = null;
      try
      {
        xmlFormat = new XmlSerializer(typeof(Player));
      }
      catch (Exception ex)
      {
        Debug.Write(ex.InnerException.ToString());
      }

      if (store.FileExists("Gracz.xml"))
      {
        var fs = store.OpenFile("Gracz.xml", FileMode.Open);

        using (StreamReader sw = new StreamReader(fs))
        {
          return (Player)xmlFormat.Deserialize(sw);
        }
      }
      else throw new Exception("Brak pliku z zapisem gry");


    }

    public void SavePlayer()
    {
      var store = IsolatedStorageFile.GetUserStoreForApplication();
      XmlSerializer xmlFormat = null;
      try
      {
        xmlFormat = new XmlSerializer(typeof(Player));
      }
      catch (Exception ex)
      {
        Debug.Write(ex.InnerException.ToString());
      }

      if (store.FileExists("Gracz.xml"))
      {
        store.DeleteFile("Gracz.xml");
      }

      var fs = store.CreateFile("Gracz.xml");
      using (StreamWriter sw = new StreamWriter(fs))
      {
        xmlFormat.Serialize(sw, this);
```

```
        }


        if (store.FileExists("Gracz.xml"))
        {
            var fss = store.OpenFile("Gracz.xml", FileMode.Open);
            using (StreamReader sr = new StreamReader(fss))
            {

                string xmls = sr.ReadToEnd();

                Debug.Write(xmls);
                store.Close();
            }
        }
        else
        {
            Debug.Write("Plik nie istnieje");
                    throw new Exception("Brak pliku z zapisem gry");
        }
    }
}
```

Rysowanie postaci na mapie odbywa się za pomocą metody Draw, która na podstawie podanej pozycji, ilości klatek, szablonu postaci oraz aktualnej klatki oblicza co ma zostać wyrysowane na mapie.

Sterowanie animacją postaci realizowane jest przy pomocy metody Move, która przyjmuje typ wyliczeniowy „Direction”, zmieniający się odpowiednio podczas naciskania klawiszy sterujących, oraz listę tekstur. Na tej podstawie zmienia aktualnie wyświetlaną teksturę przez metodę Draw.

```
public void Move(Direction direction, List<Texture2D> tx)
    {
        switch (direction)
        {
            case Direction.Idle_Down:
            { IsMoving = false; Texture = tx [4]; break; }
            case Direction.Up:
            { IsMoving = true; Texture = tx [2]; break; }
            case Direction.Down:
            { IsMoving = true; Texture = tx [3]; break; }
            case Direction.Left:
            { IsMoving = true; Texture = tx [1]; break; }
            case Direction.Right:
            { IsMoving = true; Texture = tx [0]; break; }
            case Direction.Idle_Left:
            { IsMoving = false; Texture = tx [5]; break; }
            case Direction.Idle_Right:
            { IsMoving = false; Texture = tx [6]; break; }
            case Direction.Idle_Back:
            { IsMoving = false; Texture = tx [7]; break; }
            default:
            break;
        }
    }
```

Metoda symulująca głód w grze „Spadek_Glod” przyjmuje jako argument czas gry, dzięki temu możliwe jest obniżanie statystyki głodu z prędkością określoną za pomocą zmiennej Glod_Predkosc.

```
public void Spadek_Glod(GameTime gameTime)
    {
      Glod_Czas += gameTime.ElapsedGameTime.Milliseconds*20;

      if (Glod_Czas > Glod_Predkosc)
      {
        Glod_Czas -= Glod_Predkosc;
        if (GLOD != 0)
        { GLOD -= 1; }
        if (GLOD < 20)
        {
         try
         {
           materials.Food -= 5;
         }
         catch (RawMaterials.OutOfFoodException ex)
         {
        Game1.messages.AddMessage("jesteś głodny”,new Rectangle(10,10,100,100));
         HP -= 1;
         }
        }
        Glod_Czas = 0;
      }
    }
```

Klasa RawMaterials

Klasa opisuje surowce dostępne w grze. Są to drewno, kamień, liany, metal, woda oraz jedzenie. Posiada ona własne wyjątki opisujące brak danych surowców.

Jedna z właściwości tej klasy:

```
public int Food
    {
      get { return _food; }
      set {
        if (_food + value < 0)
        {
          throw new OutOfFoodException("Zbyt mało jedzenia!");
        } else
        {
          _food = value;
        }
      }
    }
```

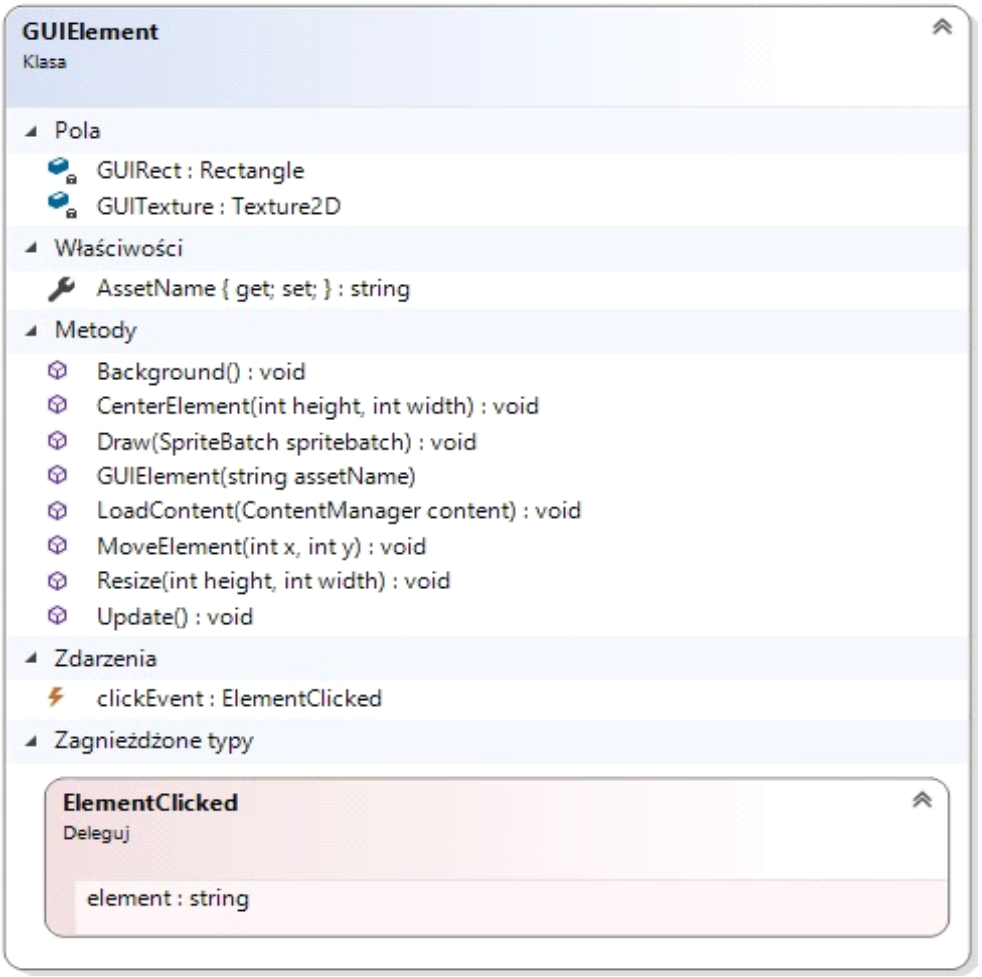
Klasa Tool

Klasa opisuje narzędzia, które użytkownik może tworzyć w trakcie rozgrywki. Zawiera informacje o obiekcie **RawMaterials** który opisuje wymagania potrzebne do stworzenia przedmiotu oraz tablicę obiektów **Tool** która stanowi o innych narzędziach potrzebnych do utworzenia danego przedmiotu.

Menu główne

Menu główne działa w oparciu o dwie klasy. Pierwsza GUIElement służy do opisu poszczególnych części menu, tj. tło, logo, przyciski. Konstrukcja głównego menu zakłada łatwą rozszerzalność o dodatkowe funkcje.

GUIElement



Klasa ta opiera się na dwóch polach tworzących pojedynczy element (stąd też nazwa) menu. Pierwsze z nich określa obszar a drugi teksturę. Oprócz pól każdy element ma również przypisane zdarzenie oraz delegat.

Metody Klasy GUIElement:

public void LoadContent(Content Manager content) – metoda ustawiająca domyślne wartości pół elementu, tj. ładuje texturę z pliku oraz przypisuje początkowy obszar (prostokąt) który będzie zajmował element.

public void Update() – metoda sprawdzająca czy dotknięty został obszar elementu (prostokąta) i jeśli tak, wywołuje zdarzenie clickEvent.

public void Draw(SpriteBatch spritebatch) – metoda odpowiedzialna za narysowanie elementu.

public void CenterElement(int height, int width) – metoda służaca do wyświetlania elementów na środku ekranu.

public void Resize (int height, int width) – metoda pomocnicza pozwalająca manipulację szerokością i wysokością elementu.

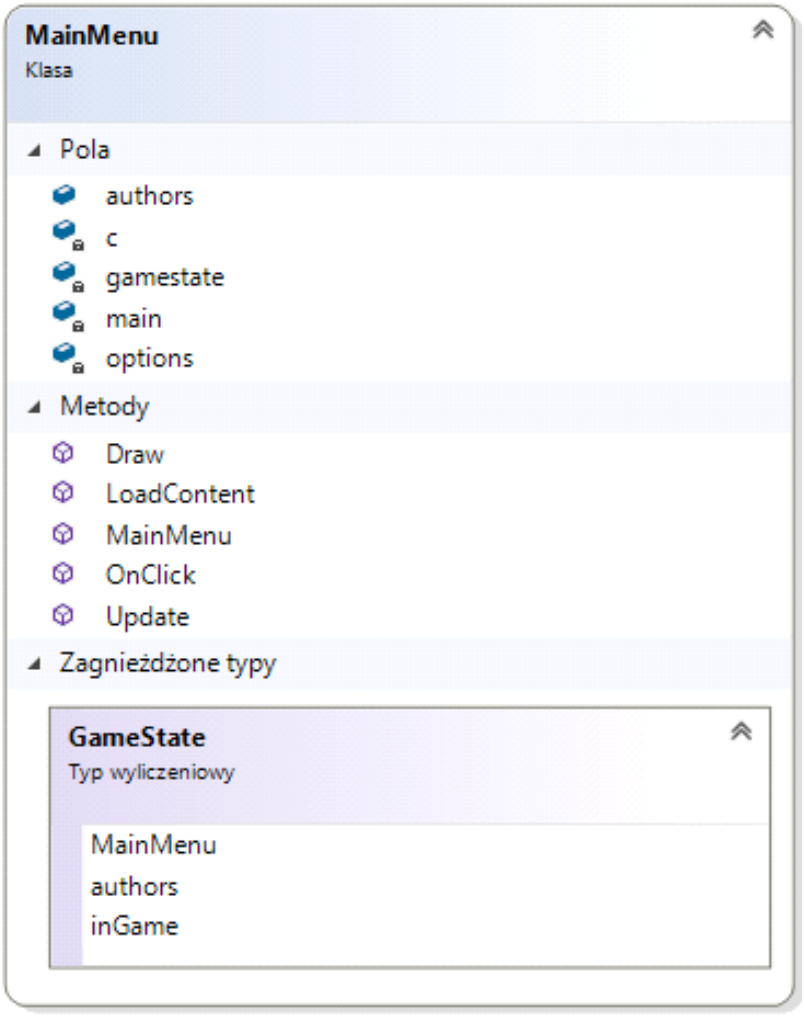
public void Background() – metoda ustawiająca element jako tło, tj. rozciągająca go odpowiednio do wielkości ekranu

public void MoveElement(int x, int y) – przesuwa element bez zmiany jest wymiarów

public event ElementClicked clickEvent

public delegate void ElementClicked(string element) – delegat do późniejszego wykorzystania przy implementacji reakcji na naciśnięcie

MainMenu



Klasa zarządzająca kontenerami obiektów typu GUIElement. Poruszanie się po menu głównym polega na zmianie stanu instancji typu wyliczeniowego „GameState" widocznego powyżej. W zależności od stanu wyświetlane są określone obiekty klasy GUIElement.

public void LoadContent(Content Manager content) – metoda wywołująca wczytywanie tekstur dla każdego elementu w poszczególnych kontenerach oraz odpowiednio je rozmieszczająca za pomocą funkcji CenterElement, MoveElement, Background oraz Resize

public void Update() – metoda sprawdzająca w zależności od GameState kliknięcie na przycisk.

public void Draw(SpriteBatch spritebatch) – metoda odpowiedzialna wywoływanie rysowania poszczególnych elementów w zależności od GameState'u

public void OnClick(string element) – metoda dodawana do delegatu określająca zachowanie aplikacji po naciśnięciu na przycisk. Rozróżnianie jest to, czy aktualnie jesteśmy w menu głównym czy w submenu.

Klasa Map

Klasa odpowiedzialna za przechowywanie informacji o mapie jak i samej map ,umożliwia zmianę pozycji mapy (ponieważ zdecydowaliśmy się na wariant w którym gracz jest rysowany na środku ekranu a to mapa się przesuwa.

Mapa gry składa się z 5 warstw i każda warstwa posiada siatkę pól do których są przypisane ID. Pierwsza warstwa przechowuje informacje o podłożu na którym stoi gracz. Druga warstwa są to obiekty to obiekty które znajdują się na podłoży czyli na wysokości gracza ,a w trzeciej warstwie są obiekty znajdujące się ponad graczem i za którymi może przejść. Czwarta warstwa służy do identyfikacji obiektów i jest niewidoczna dla gracza. W późniejszym etapie tworzenia gry postanowiliśmy dodać piątą warstwę która miała służyć do obsługi i identyfikacji misji (umieszczanie znaczników na mapie).

Klasa ta korzysta z listy klas MapTexture oraz tablicy dwuwymiarowej Tile które odpowiadają za:

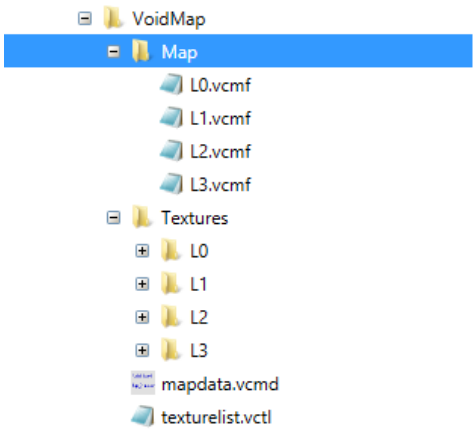
Klasa MapTexture przechowuje teksturę informacje do której warstwy ona przynależy oraz jakie ID posiada.

Klasa Tile odpowiada za przechowywanie informacji o jednym polu jednej warstwy mapy. Rozmiar tablicy zależy od rozmiarów mapy.

W klasie tej jest także obliczany rozmiar jednego pola mapy
`private int MapZoom = ((ScreenHeight / 10) + (ScreenWidth / 17)) / 2;`
Gdzie wartości 10 i 17 to ilość pól które są widoczne na ekranie odpowiednio w wysokości i szerokości okna.

```
private void GetMapSize() {
            using (var stream = TitleContainer.OpenStream("Content/Maps/" + MapName +
                                                          "/mapdata.vcmd"))
    using (var reader = new StreamReader(stream)) {
       this.MapName = reader.ReadLine();
       this.Width = int.Parse(reader.ReadLine());
       this.Height = int.Parse(reader.ReadLine());
       this.NumberOfLayers = int.Parse(reader.ReadLine());
    }
}
```

Przykładowa metod wczytująca dane z pliku mapy (mapdata.vcmd)



Struktura katalogów mapy (w katalogach od L1 do L3 znajdują plik **.png* z grafikami). Struktura ta jest tworzona przez kreator map ,który napisaliśmy aby przyśpieszyć prace przy projektowaniu większych map.

Plik *texturelist.vctl* zawiera listę tekstur opisanych jako warstwa i identyfikator do których należą oraz nazwę tekstury i ścieżkę do niej (tekstury są kopiowane przez kreator map do katalogu *Textures/Lx)*

Plik *mapdata.vcmd* to plik zawierający nazwę ,rozmiary mapy i ilość warstw. Może on posłużyć do wczytania tego projektu i kontunuowania prac (można go także przypisać do aplikacji kreatora i uruchamiać).

Plik *Map/Lx.vcmf* odpowiada za przechowywanie tablicy dwuwymiarowej identyfikatorów pól mapy z warstwy ‘x’ oddzielonych spacjami

Klasa Messages

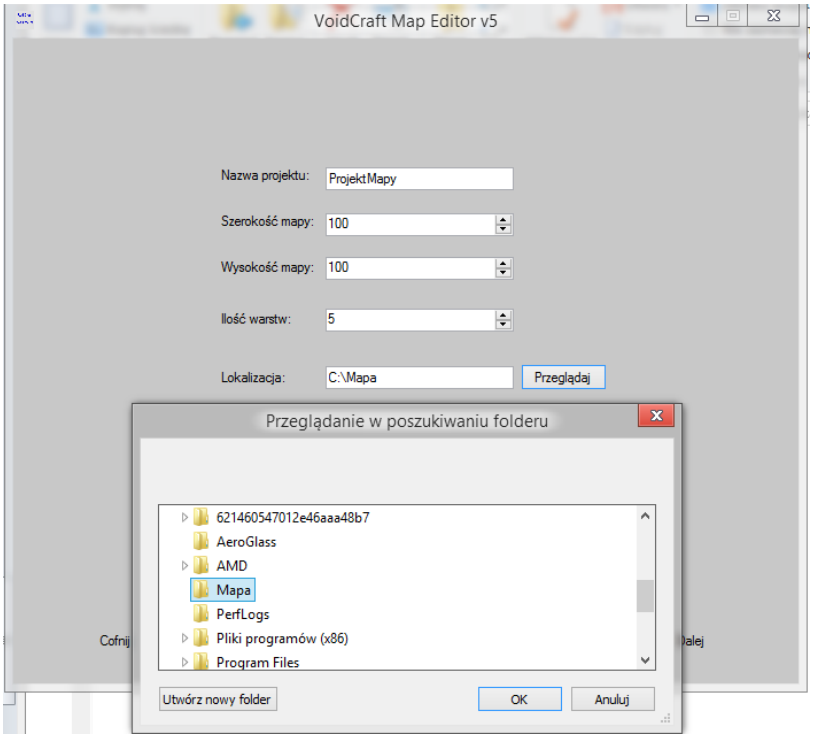
Klasa umożliwia wyświetlanie 2 typów komunikatów dla użytkownika . Pierwszy typ to komunikaty które wyświetlają się według kolejności wykonanych akcji ,a drugi to komunikaty które mogą być niezależnie wyświetlane i umieszczane w dowolnym miejscu ekranu.

Klasa umożliwia dodanie wiadomości (samego tekstu) lub podanie własnego pliku graficznego. Aktualnie istniej limit do trzech ostatnie wiadomości ,wiadomości starsze zostają usunięte. Komunikaty można usunąć poprzez kliknięcie w obszar danej wiadomości.

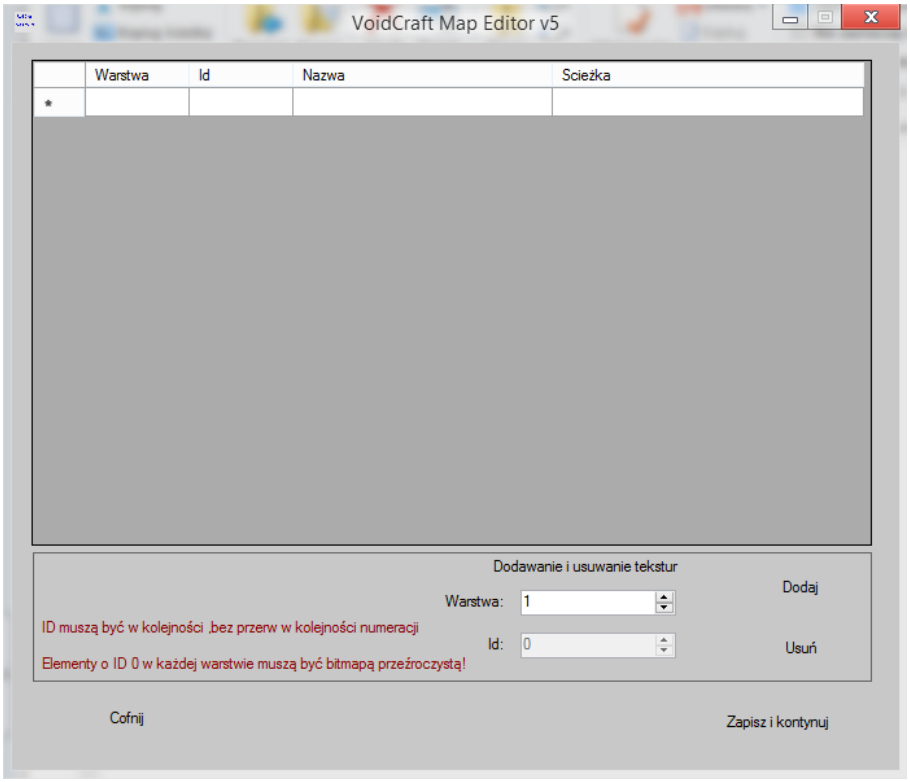
Kreator map

Jest to narzędzie które stworzyliśmy aby przyśpieszyć prace przy projektowaniu mapy gry (katalogi i pliki zostały omówione w ramach klasy MapController).

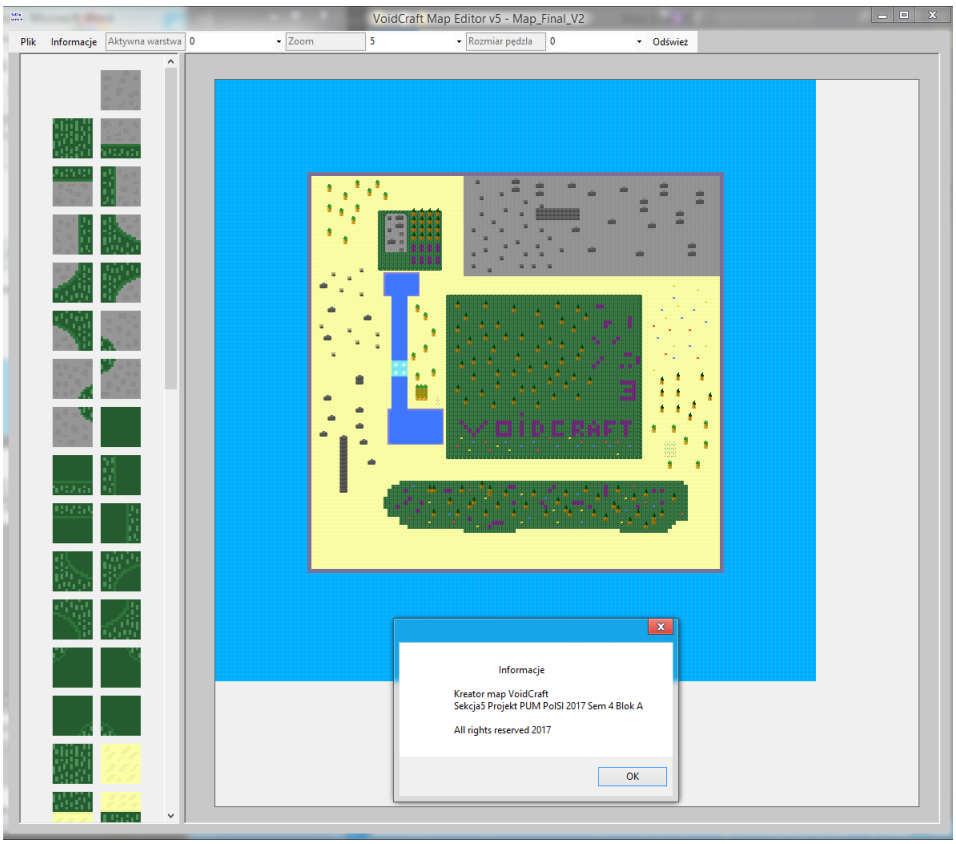
Widok programu po uruchomieniu



Tworzenie nowego projektu i wybieranie lokacji zapisu



Okno dodawania tekstur do warstw



Widok głównej części programu ,po lewej lista tekstur dla aktywnej warstwy ,po prawej okno w którym umieszczamy tekstury do określonych warstw.

Program umożliwia zapis projektu mapy do aktywnego projektu lub nowego ,utworzenie nowego projektu oraz umożliwia poniższe funkcje (wraz ze skrótami klawiszowymi)

- Control + A - Wklejenie na całej mapie tekstury wybranej LPM
- Alt + A - Wklejenie na całej mapie tekstury wybranej PPM
- Control + S - Zapis projektu
- Control + Z - Przełączenie linii siatki
- Control + X - Przełączenie numeracji linii
- Control + C - Przełączenie wskaźnika obszaru
- Control + Space - Przełączenie automatycznego odświeżania
- Space - Wymuszenie odświeżenia
- Q - Zoom +
- W - Zoom -
- D - Zmiana aktywnej warstwy +
- F - Zmiana aktywnej warstwy -
- LPM - Wklejenie wybranej lewym przyciskiem tekstury na wybrany obszar
- PPM - Wklejenie wybranej prawym przyciskiem tekstury na wybrany obszar
- ŚPM - Wklejenie wybranych LPM i PPM tekstur na wybrany obszar w formie [L][P]

4) Zastosowane narzędzia

- Microsoft Visual Studio Enterprise 2015
- Biblioteki MonoGame
- Język C#
- System kontroli wersji GitHub

5) Podsumowanie

a) Harmonogram oraz podział prac

Alan Biegun – twórca kreatora map oraz opracowanie poruszania się po mapie
Piotr Zuber – planista, opracowanie systemu craftingu
Konrad Sladkowski – twórca głównego menu
Jan Snopek – grafik i animator, twórca klasy gracza
Dawid Ryl - grafik, twórca projektu mapy i menu w grze.

b) Wnioski

Realizacja tego projektu uświadomiła nam, jak wiele możliwości otwiera podejście obiektowe w programowaniu. Szczególny nacisk położono na staranne rozplanowanie prac oraz komunikacje w zespole, co mimo wielu trudności z czasem zaowocowało.

Projekt przyniósł wraz z wieloma wyzwaniami tyle samo doświadczeń. Co prawda nie tylko tych przyjemnych aczkolwiek na pewno przydatnych w przyszłej pracy zawodowej. Przejrzysty podział obowiązków i ról, ustalanie terminów „deadline” czy wzajemna pomoc nauczyły nas, że komunikacja w zespole jest równie ważna, jak umiejętności programistyczne.

c) Spis literatury

- Andrew Troelsen, “Język C# 2010 i platforma .NET 4”
- Serwis internetowy http://www.monogame.net/documentation/?page=main