UNIWERSYTET WARMIŃSKO-MAZURSKI

W OLSZTYNIE

WYDZIAŁ MATEMATYKI I INFORMATYKI

Kamil Zieliński

Kierunek: Informatyka

Realizacja gry 2D z gatunku strategicznych gier czasu rzeczywistego w silniku Unity

Praca inżynierska wykonana
w Katedrze Analizy Zespolonej
pod kierunkiem
dr. Jacka Marchwickiego

Olsztyn 2022

UNIVERSITY OF WARMIA AND MAZURY IN OLSZTYN

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

Kamil Zieliński

Field of study: Computer Science

Development of 2D real-time strategy game in Unity engine

Engineering Thesis written in
Chair of Complex Analysis
under supervision of
dr Jacek Marchwicki

Olsztyn 2022

Streszczenie

Praca zawiera dokumentację na temat strategicznej gry czasu rzeczywistego „Bulwarriors” w 2D zbudowanej na silniku Unity. Projekt został stworzony na potrzeby realizacji pracy inżynierskiej. Dokumentacja zawiera opis gatunku jakim są gry RTS (Real-Time Strategy), użytych narzędzi z którymi aplikacja została zaprojektowana oraz przede wszystkim szczegółowy opis samej gry, składający się z objaśnienia rozgrywki, czyli mechanik występujących w grze, wyjaśnienia sterowania, przedstawienie stworzonego świata gry oraz sposobie implementacji poszczególnych funkcji systemu.

Abstract

This paper contains documentation about 2D real-time strategy game called "Bulwarriors" built in Unity engine. Project was created for the purpose of elaborate an engineering thesis. Documentation contains a description of RTS game genre (Real-Time Strategy), tools used for application design and most of all - detailed description of the game itself consisting gameplay clarification, mechanics occurring in the game, control details, presentation of the game world and how to implement various features of the system

Spis treści

1. Wprowadzenie

Celem pracy inżynierskiej jest przedstawienie projektu oraz implementacja gry strategicznej czasu rzeczywistego w widoku dwuwymiarowym. Polegającej na pokonaniu przeciwnika równocześnie obronie przed jego atakami. Gracz na początku rozgrywki posiada pewną ilość jednostek oraz budynków za pomocą, których musi zacząć rozwijać swoją siedzibę, w celu zgromadzenia większych sił w postaci jednostek bojowych oraz budowie kolejnych struktur umożliwiających zbieranie zasobów, jak również tworzeniu lepszych oddziałów. Zwycięstwo zostanie osiągnięte w momencie zniszczenia wszystkich budynków strony przeciwnej. Oponent dysponuje jednostkami oraz budynkami tego samego typu co gracz. Przeciwnik, sterowany stworzonym algorytmem próbuje wyeliminować gracza niszcząc wszystkie postawione należące do niego budynki, powodując porażkę.

Aplikacja została zaimplementowana z użyciem silnika Unity 2D, oprogramowania GIMP 2.8, git oraz języka programowania C#. Projekt był inspirowany znanymi tytułami, o których mowa w dalszej części pracy, jak również własnymi pomysłami na świat gry oraz implementacje mechanik.

Dokumentacja zawiera opis gatunku jakim są gry RTS w dodatku tytuły, którymi był inspirowany projekt, użytych narzędzi z którymi aplikacja została zaprojektowana oraz przede wszystkim szczegółowy opis samej gry, składający się z objaśnienia rozgrywki, czyli mechanik występujących w grze, wyjaśnienia sterowania oraz sposobu implementacji poszczególnych funkcji.

2. Opis gatunku

2.1. RTS jako gatunek gier

Gra RTS oznacza gatunek „strategicznych gier czasu rzeczywistego” (ang. Real-time strategy), w których gracz podejmuje decyzje w celu osiągnięcia warunków zwycięstwa lub zrealizowania jakiegoś założonego celu. Gry tego typu często stawiają na szybkość podejmowania strategicznych decyzji, pozwalających zyskać przewagę nad przeciwnikiem, zwiększają szanse przetrwania lub wpływają korzystnie na wynik końcowy. Niezbędna szybkość podejmowanych działań jest narzucona ze względu, że gra trwa w czasie rzeczywistym, nie jest ograniczona turami, gdzie gracze mają pewną ilość czasu na podjęcie kluczowych decyzji. W tym trybie, każdy gracz w tym przeciwnik wykonuje ruch lub podejmuje jakąś decyzję w tym samym czasie.

2.2. Przeznaczenie

W gatunku RTS, sporo osób może znaleźć element, którym się zainteresuje. Przez budowanie olbrzymich siedzib, zbieraniu surowców, stworzeniu ekonomicznej potęgi, czy niezwyciężonych fortec, poprzez tworzenie armii zdolnej niszczyć przeciwników i zdobywaniu terytorium do dalszej ekspansji.

Sama mechanika bywa dla graczy satysfakcjonująca, dodatkowo nietrudno dodać do gier RTS fabułę, czy swego rodzaju kampanię, która może dodatkowo urozmaicić grę, wprowadzając kolejny cel i pozwalając na satysfakcjonującą wielogodzinną rozgrywkę.

Wiele gier posiada podsumowania rozegranych potyczek, zawierającej spis zebranych zasobów czy jednostek w czasie, co stanowi satysfakcjonujące podsumowanie całej sesji.

2.3. Znane tytuły

2.3.1. StarCraft

Cała seria gier elektronicznych stworzona przez studio Blizzard Entertainment, seria jest jedną z najpopularniejszych gier RTS. Wydana w 1998 roku pierwsza część, stała się jedną z najbardziej popularnych gier tego gatunku.

StarCraft posiadał zaimplementowane funkcje, które były standardem dla tego typu gier. Gra wyróżniła się ze względu na urozmaicenie gry wynikające z wprowadzenia różnych ras, różniących się sposobem gry oraz funkcjami. Wymagało to od gracza zmianę strategii w zależności do wybranej nacji[1].

Rysunek 1 fragment rozgrywki podczas walki dwóch nacji z gry StarCraft[2]

Wydany w 2010 StarCraft II był kolejną odsłoną gry, będącą ulepszeniem poprzedniej części. Produkcja cieszyła się większym zainteresowaniem od poprzedniego wydania, szczególnie w branży sportu internetowego[3].

[1] https://www.britannica.com/topic/StarCraft
[2] http://videogamesandbooze.blogspot.com/2010/
[3] https://www.ign.com/articles/2016/03/22/the-rise-and-fall-of-starcraft-ii-as-an-esport

Rysunek 2 fragment rozgrywki z gry StarCraft[4]

Gracz rozpoczyna grę z grupą robotników umożliwiających zbieranie zasobów, czyli kryształów i gazu, jak również tworzenie nowych struktur, zapewniających produkcje różnego typu jednostek oraz ulepszanie ich, zwiększając poszczególne statystki danego rodzaju jednostki. Zasoby pełnią rolę waluty, za pomocą której gracz może zlecić budowę, ulepszenie czy produkcję.

Celem gry jest wyeliminowanie przeciwnika, którego celem jest doprowadzenie do przegranej gracza.

Szata graficzna, mechaniki oraz kampania gry stanowiły odpowiednią symulację świata gry stworzonego w scenerii wojen kosmicznych.

[4] https://allegro.pl/artykul/starcraft-2-battle-chest-recenzja-gry-LvKnem1VLia

2.3.2. The Lord of the Rings: The Battle for Middle-Earth

Gra stworzona przez EA Los Angeles w 2004 roku, jest oparta na filmowej trylogii Władca Pierścieni w reżyserii Petera Jacksona, która z kolei jest na podstawie powieści J.R.R. Tolkiena oryginalnej powieści.

Produkcja jest grą gatunku strategii czasu rzeczywistego, gracze jak również przeciwnicy gromadzą zasoby, wykorzystując je do budowy baz wojskowych i armii. Struktury są budowane na z góry ustalonych działkach na mapie gry. W grze jest jeden zasób, produkowany za pośrednictwem dedykowanych budynków[5].

Rysunek 3 fragment rozgrywki z gry The Lord of the Rings: Battle for Middle-Earth[6]

Armia jaką gracz dysponuje składa się z grup jednostek, jak również pojedynczych, gdzie każda posiada swoje statystyki. Frakcje do dyspozycji posiadają różne typy jednostek, mniej lub bardziej różniące się założeniami i mechaniką.

Bohaterowie są specjalnym rodzajem pojedynczej jednostki, jest ona wyposażona w szereg najróżniejszych umiejętności, zwiększających tymczasowo siłę jednostki czy zadających duże obrażenia.

[5] https://en.wikipedia.org/wiki/The_Lord_of_the_Rings:_The_Battle_for_Middle-earth
[6] https://www.moddb.com/mods/bfme-patch-108/images/rohan

3. Założenia techniczne

3.1. Platforma

Gra została zaprojektowana na system Windows, który jest dominującym system operacyjnym. Globalnych udział w rynku systemów operacyjnych wśród komputerów stacjonarnych wynosi ponad 70%[7]. Co więcej system Windows jest kompatybilny z większością współczesnych gier, dlatego posiada go większość użytkowników.

Problematycznymi wydają się platformy mobilne czy konsole, ze względu na charakterystykę sterowania, która w tych przypadkach jest utrudniona. Sterowanie w tego typu grze jest znacząco uproszczone w przypadku użycia myszy komputerowej.

Wybierając platformy powinno się uwzględnić przede wszystkim grupę użytkowników, która najchętniej zainteresuje się tytułem. Dlatego w przypadku tego projektu został wybrany system Windows.

3.2. Unity

Aplikacja została zaimplementowana z użyciem silnika Unity w wersji 2020.3.20f1 przy pomocy komponentów potrzebnych do stworzenia gry 2D.

Silnik Unity został opracowany przez Unity Technologies, produkt został wypuszczony w czerwcu 2005 roku i od tamtego czasu jest stale rozwijany i aktualizowany.

Silnik służy do wszechstronnych projektów zarówno 2D jak i 3D, poza obsługą elementów fizycznych, pozwalających na tworzenie mechanik gry, umożliwiona jest również opcja tworzenia animacji na samej platformie Unity jak i prostych modeli 3D[8].

[7] https://www.statista.com/statistics/268237/global-market-share-held-by-operating-systems-since-2009/
[8] https://en.wikipedia.org/wiki/Unity_(game_engine)

3.3. C#

C# język programowania jest głównie używany w silniku unity do pisania skryptów do aplikacji. Istnieją również wtyczki oraz pakiety, które umożliwiają pisanie skryptów w innych językach.

Do tworzenia skryptów w języku C# użyte zostało zintegrowane środowisko programistyczne Microsoft Visual Studio 2019 firmy Microsoft.

3.4. GIMP

Narzędziem użytym do tworzenia oraz obrabiania grafik użytych w grze posłużył program GIMP 2.8.18. Program jest edytorem grafiki rastrowej na licencji wolnego i otwartego oprogramowania (General Public License)[9].

3.5. GitHub

GitHub – jest serwisem internetowym przeznaczonym do projektów programistycznych, wykorzystujący system kontroli wersji Git.

Git jest systemem kontroli wersji, niemal niezbędnym w realiach dzisiejszej pracy z kodem oraz aplikacjami. Najważniejszym zastosowaniem oprogramowania git jest możliwość efektywnej pracy zespołowej na jednym projektem programistycznym.

W tym projekcie mimo pracy w jednoosobowym zespole, był niezbędny ze względu na łatwą i bezpieczną ingerencje w stworzony projekt. Pozwalającą na pracę na tak zwanych gałęziach, które stanowią pewnego rodzaju kopię projektu oraz historię zmian na danej gałęzi co umożliwia łatwy powrót w przypadku popełnienia jakiegoś błędu i chęci powrotu do poprzedniej wersji.

[9] https://www.gimp.org/about/

4. Rozgrywka

4.1. Interakcja z otoczeniem

Gracz w ramach sterowania obiektami będzie mógł sterować jednostkami oraz budynkami, które są do niego przydzielone. Przeciwnik dysponuje tymi samymi rodzajami obiektów. Gracz może zlecić swoim jednostkom różne polecenia, takie jak:

- Przemieszczenie jednostek
- Atak na jednostkę przeciwnika
- Atak na budynek przeciwnika
- Budowa struktury

Budynki gracza dysponują swoimi funkcjami aktywnymi oraz pasywnymi. Do aktywnych należy rekrutacja jednostek, natomiast do pasywnych dodawanie złota.

4.2. Przebieg rozgrywki

Gracz zaczyna z jednym budynkiem głównym „Cytadelą” oraz grupą „podstawowych robotników” oraz z pewną ilością złota. Wraz z czasem budynek generuje przychód, który pozwala na budowę struktur przy pomocy robotników, które pozwalają na zwiększenie zarobków oraz rekrutacje jednostek.

Przeciwnik w tym czasie również wykonuje swoje ruchy w tym wykonując co pewną ilość czasu atak grupą jednostek na bazę gracza usiłując ją zniszczyć.

4.3. Cel gry

Wymiana ataków w pewnym momencie doprowadzi do zyskania przewagi jednej ze stron, która doprowadzi do zniszczenia budynków przeciwnika, następnie gra się kończy, wyświetlając statystki gracza, informując czy wygrał lub przegrał.

4.4. Podsumowanie rozgrywki

Rozgrywka kończy się w momencie zniszczenia wszystkich struktur jednej ze stron. Skutkuje to pojawieniem się ekranu zwycięstwa lub przegranej. Gracz ma możliwość wyjścia do menu głównego oraz wyświetlenia szczegółowych statystyk z gry.

Szczegółowe statystyki są pokazane w postaci wykresu określającego, ile gracz posiadał jednostek, złota lub punktów w określonym czasie.

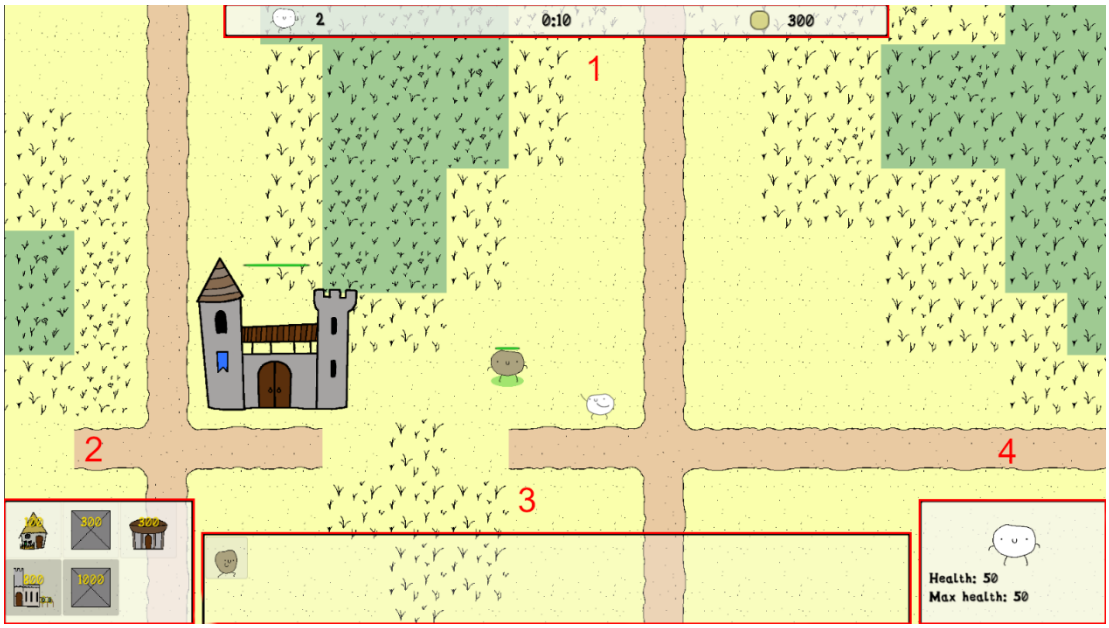
Rysunek 4 ekran po zakończonej rozgrywce z wyświetlonym wykresem zależności między złotem a czasem rozgrywki

Liczba punktów jest obliczana na podstawie ilości jednostek, zdobytych zasobów oraz czasu.

$$punkty = \frac{liczbaJednostek * modyfikatorJednostki + liczbaZlota}{czas}$$

- liczbaJednostek – liczba określająca, ile gracz zdobył jednostek
- modyfikatorJednostek – modyfikator określający jak cenne są jednostki dla punktacji
- liczbaZłota – liczba określająca, ile gracz zdobył zasobów
- czas – czas trwania rozgrywki, wyrażony w sekundach

4.5. Opis UI



Rysunek 5 fragment rozgrywki z wyszczególnionymi elementami interfejsu gracza

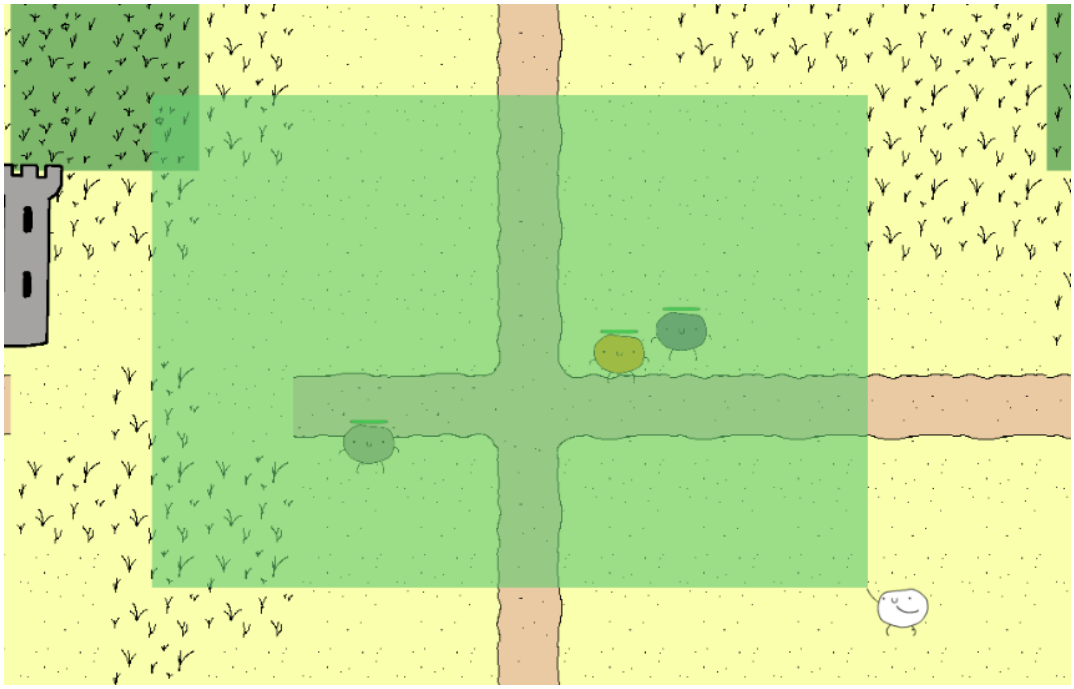
1. Ramka zasobów fragment UI przedstawiający aktualne ilości jednostek, złota oraz czas trwania rozgrywki.
2. Ramka akcji zawiera przyciski akcji, które się wyświetlają, jeżeli zostanie zaznaczona jednostka lub budynek zawierający akcje związane z rekrutacją lub budowaniem.
3. Ramka listy przedstawia wszystkie zaznaczone jednostki w postaci miniaturki.
4. Ramka statystyk przedstawia szczegółowe statystyki pojedynczego obiektu. W przypadku zaznaczenia wielu obiektów, nic nie będzie w niej wyświetlane.

4.6. Sterowanie

4.6.1. Mysz komputerowa

Sterowanie opiera się głównie na myszy komputerowej, z jej pomocą będzie można wykonywać większość operacji.

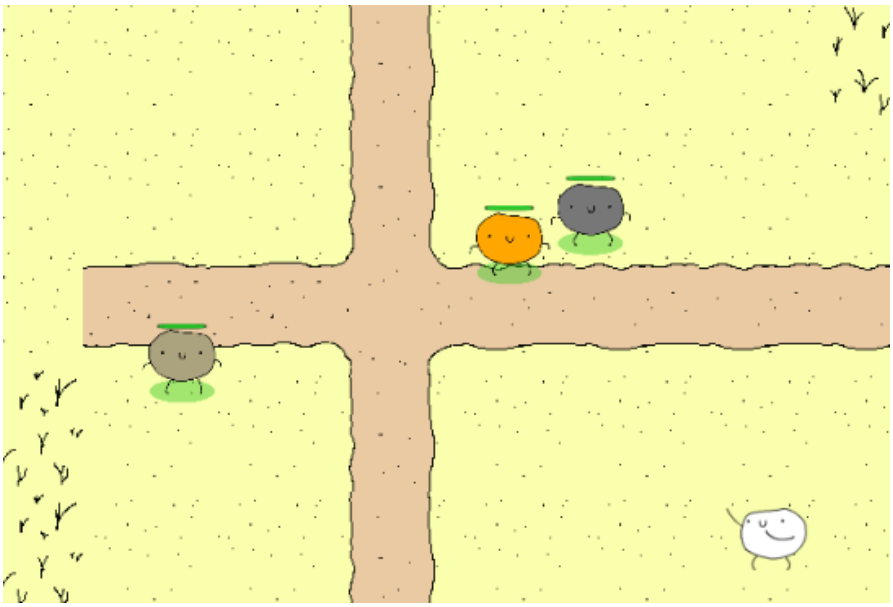
4.6.1.1. Zaznaczanie obiektów



Rysunek 6 wizualizacja zaznaczania jednostek w grze

Zaznaczanie obiektów, polega na wciśnięciu lewego klawisza myszy, przytrzymanie go i przeciągnięcie, pojawi się zielone pole, które oznacza fragment, w którym obiekty zostaną zaznaczone, po puszczeniu lewego przycisku myszy. W sytuacji, kiedy w obrębie pola znajdą się jednostki oraz budynki, wówczas tylko jednostki zostaną zaznaczone. Jeżeli tylko pojedynczy budynek znajdzie się w polu to zostanie zaznaczony. Nie ma możliwości zaznaczenia kilku budynków.

4.6.1.2. Przemieszczanie jednostek



Rysunek 7 sposób wizualizacji zaznaczonych jednostek gracza

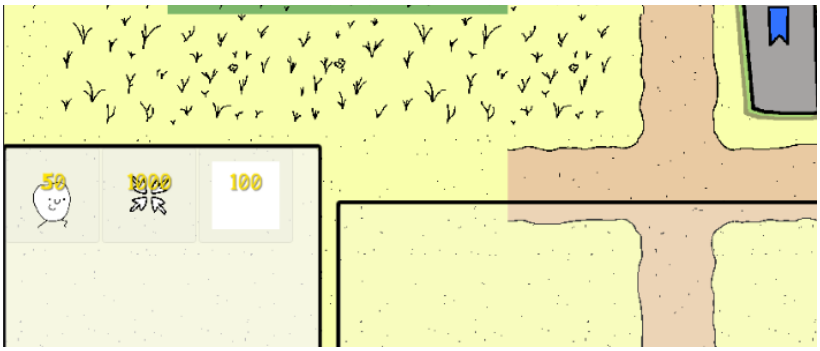
Zaznaczonym jednostkom można wydać polecenie przemieszczenia się, nakierowując kursor w wyznaczone miejsce a następnie kliknięcie prawego przycisku myszy. Jednostki przemieszczając się w wyznaczone miejsce ustawiając się obok siebie wypełniając koło wielkości zależnej od zaznaczonego oddziału.

4.6.1.3. Atakowanie oraz budowanie

Zaznaczone jednostki mogą atakować lub budować w przypadku pracowników. Komendę wydaje się poprzez najechanie kursorem na obiekt, następnie kliknięcie prawego przycisku myszy. Nakierowanie kursora na obiekty przeciwnika spowoduje wydanie polecenia ataku na budynek lub jednostkę.

Polecenie budowy nastąpi w sytuacji, kiedy wśród zaznaczonych jednostek znajdzie się co najmniej jedna jednostka budowniczych, którzy po naciśnięciu prawego klawisza myszy na budynku nie zbudowanym przemieszczą się do budynku i rozpoczną budowę.

4.6.1.4. Obsługa funkcji obiektów



Rysunek 8 fragment interfejsu zawierający przyciski

Część obiektów posiada dodatkowe funkcje pokazane w UI w sekcji „Akcji” w postaci kwadratowych guzików, które po naciśnięciu lewym przyciskiem myszy zostaną użyte. Pozwalają one na tworzenie nowych obiektów.

4.6.1.5. Przybliżanie kamery

Używając przewijania myszy istnieje możliwość przybliżania oraz oddalania kamery od obiektów gry, wprowadzenie tej funkcji pozwala graczowi na zwiększenie pola widzenia, kosztem widziany szczegółów. Granica oddalanie oraz przybliżania kamery jest z góry ustalona.

4.6.2. Klawiatura

4.6.2.1. Sterowanie kamerą

Do przemieszczania kamery przypisane są klawisze „w”, „s”, „a” oraz „d”, kolejno do przemieszczania w górę, dół, lewo, prawo. Ruch kamery jest ograniczony krawędziami.

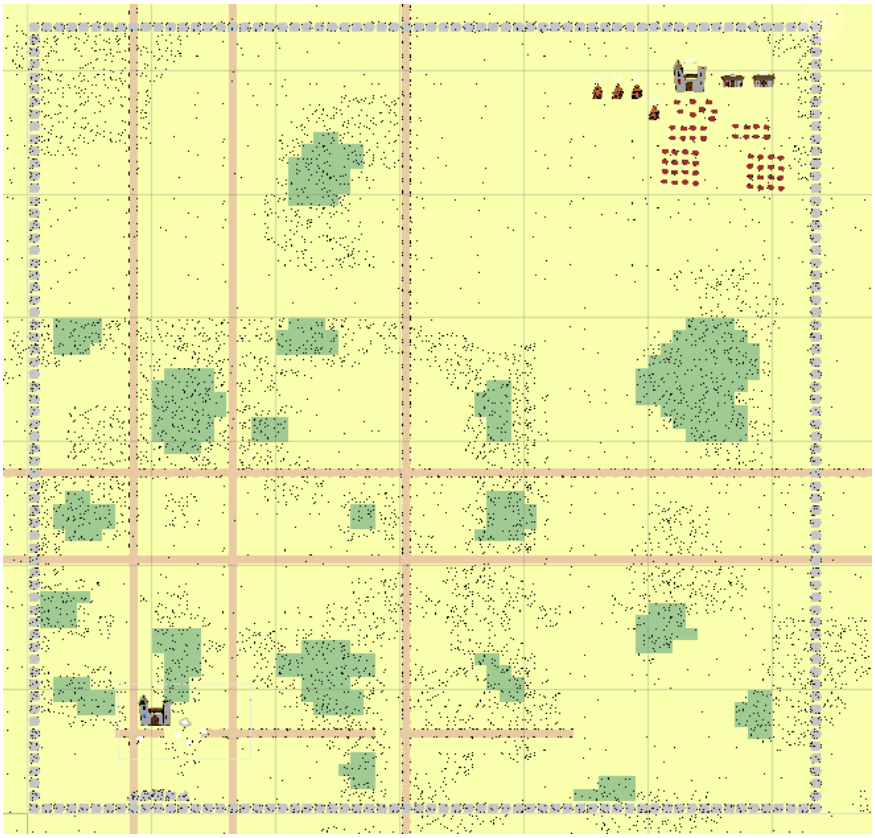
```
void Update()
{
    Vector3 position = transform.position;
    if (Input.GetKey(KeyCode.W)) position.y += cameraSpeed * Time.deltaTime;
    if (Input.GetKey(KeyCode.S)) position.y -= cameraSpeed * Time.deltaTime;
    if (Input.GetKey(KeyCode.A)) position.x -= cameraSpeed * Time.deltaTime;
    if (Input.GetKey(KeyCode.D)) position.x += cameraSpeed * Time.deltaTime;
    if(Input.GetAxis("Mouse ScrollWheel") > 0 && Time.timeScale != 0)
    {
        zoom-=0.2f;
        zoom = zoomController(zoom);
    }
    if (Input.GetAxis("Mouse ScrollWheel") < 0 && Time.timeScale != 0)
    {
        zoom += 0.2f;
        zoom = zoomController(zoom);
    }
    GetComponent<Camera>().orthographicSize = zoom;
    transform.position = new Vector3(
        Mathf.Clamp(position.x, xMinBound, xMaxBound),
        Mathf.Clamp(position.y, yMinBound, yMaxBound),
        position.z
        );
}
Odwołania: 3
private float zoomController(float zoom)
{
    if (zoom > 10f) return 10f;
    if (zoom < 0.4f) return 0.4f;
    return zoom;
}
```

Rysunek 9 fragment skryptu odpowiedzialnego za ruch kamery

4.7. Mapa

Jest to obszar, na którym toczy się rozgrywka. Gracz na mapie jest umiejscowiony w lewym dolnym rogu obszaru, przeciwnik z kolei na przeciwnej stronie, w prawym górnym rogu.

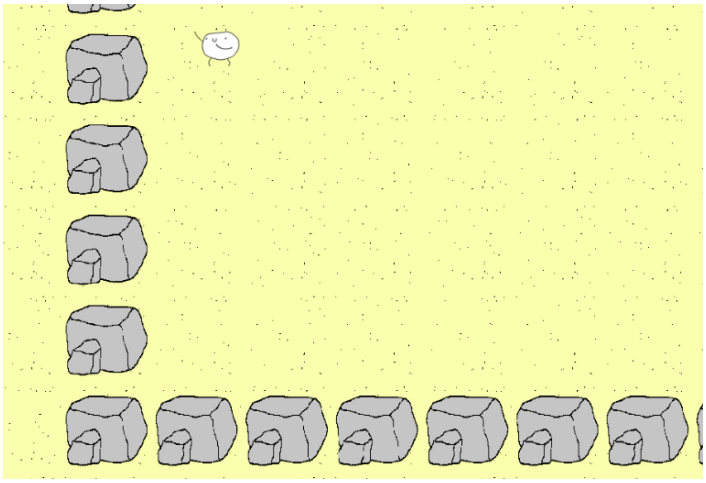
Mapa pełni funkcję wizualną, urozmaicając pole walki oraz umożliwiając orientację gracza na mapie.



Rysunek 10 mapa gry

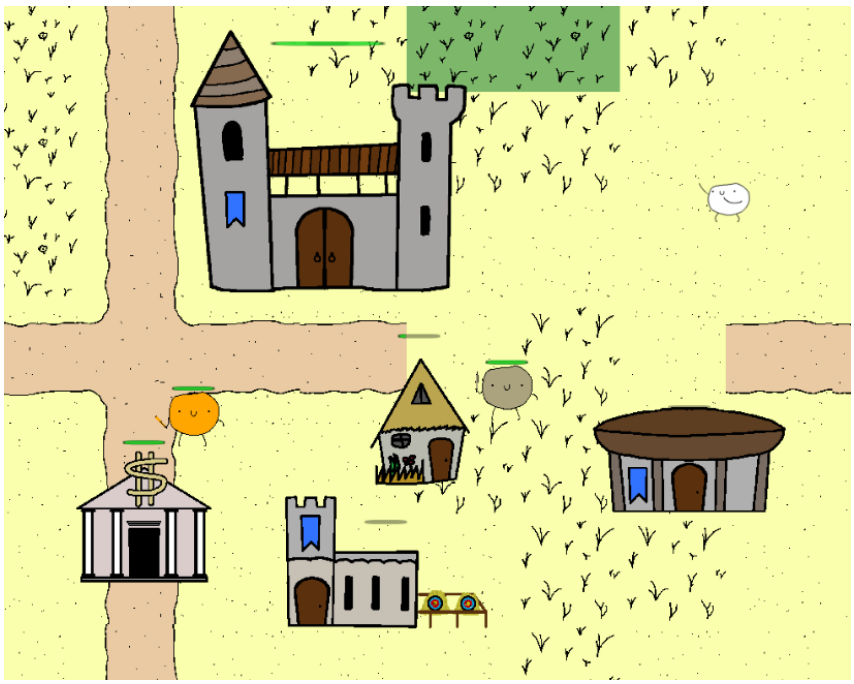
4.8. Obiekty statyczne

Kamienie – obiekt stanowiący przeszkodę, przez którą jednostki nie mogą przejść, służy do stworzenia granicy mapy. Jednostki nie mogą w żaden sposób podjąć interakcji z obiektem.



Rysunek 11 fragment granicy mapy

Budynki – stanowią obiekt statyczny, który stanowi przeszkodę, przez którą jednostki nie mogą przechodzić. Interakcje dostępne dla jednostek to budowanie w przypadku budynków sojuszniczych oraz ataku dla budynków przeciwnika.

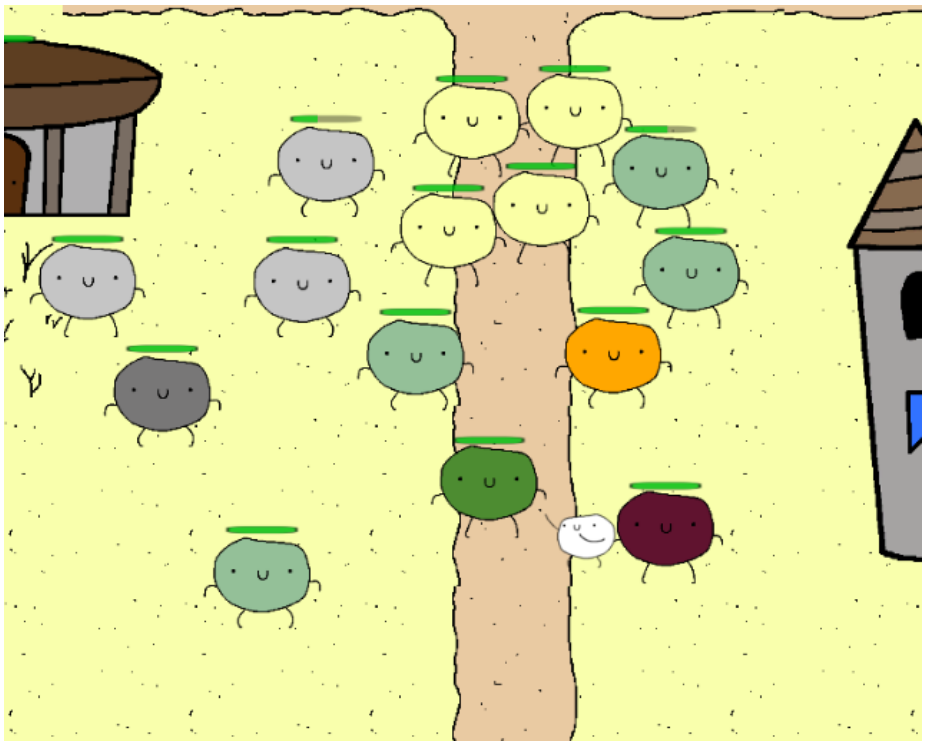


Rysunek 12 różne rodzaje budynków

4.9. Obiekty dynamiczne

Jednostki – są obiektami, które mają możliwość przemieszczania się po mapie, zarówno samoistnie, jak również za pośrednictwem komend gracza. Oddziały mają zaimplementowane animacje, aby uwydatnić funkcje jakie wykonują oraz nadać dynamikę do gry.

Jednostki mogą podejmować interakcje z niektórymi obiektami na mapie.

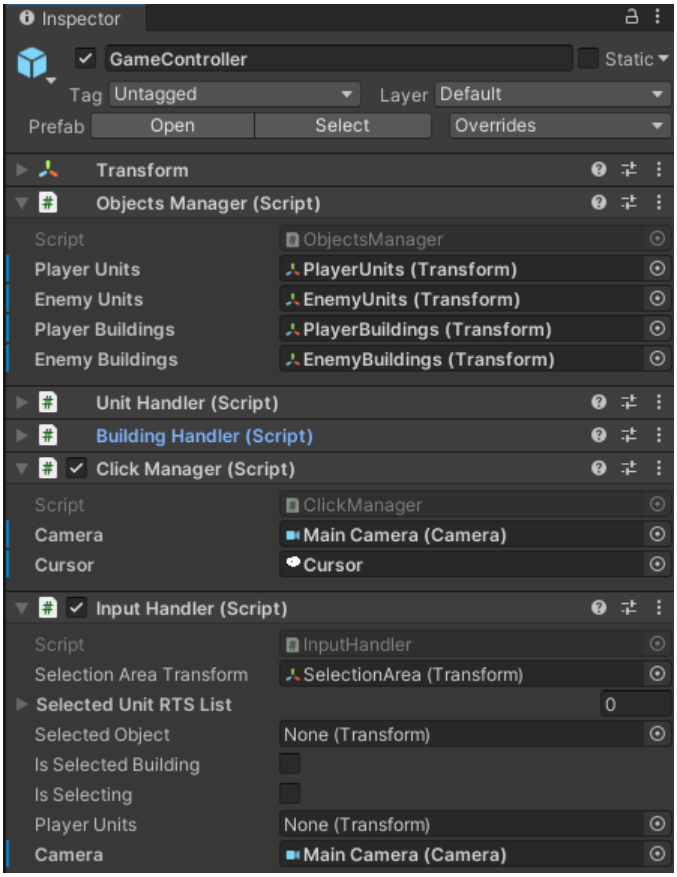


Rysunek 13 różne rodzaje jednostek

5. Implementacja

5.1. Główny kontroler

Obiekt w aplikacji odpowiedzialny za prawidłowe uruchomienie i zainicjowanie rozgrywki. Komponenty przypisują odpowiednie obiekty do gracza lub przeciwnika oraz uzupełniają statystyki obiektów znajdujących się na mapie w momencie rozpoczęcia gry na głównej scenie.



Rysunek 14 komponenty w głównym kontrolerze gry

5.2. Sterowanie obiektami

Główny kontroler zawiera skrypt odpowiedzialny za odczytywanie i obsługę urządzeń wejścia dotyczących sterowania obiektami na mapie przy pomocy zaznaczania prostokątem tworzonym na mapie od miejsca, w którym użytkownik wcisnął lewy przycisk myszy do aktualnej pozycji kursora, jeżeli klawisz jest przytrzymany, w momencie puszczenia obiekty w obszarze zostają zaznaczone, a prostokąt znika. Komponent został tak skonstruowany, aby możliwe było zaznaczenie wielu jednostek lub jednego budynku.

```
// Selection area
if (Input.GetMouseButtonDown(0))
{
    if (EventSystem.current.IsPointerOverGameObject()) return;

    foreach (Interactable interactableObject in selectedUnitRTSList)
    {
        interactableObject.gameObject.GetComponent<Interactable>().SetSelectedVisible(false);
        isSelectedBuilding = false;
    }
    if (selectedUnitRTSList.Find(x => x.GetComponent<BuildingUI>()))
    {
        selectedUnitRTSList[0].GetComponent<BuildingUI>().SetSelectedVisible(false);
        isSelectedBuilding = false;

    }
    selectedUnitRTSList.Clear();
    UIHandler.instance.UpdateSelectedUnits();

    // get cursor cordinates, when LPM state is changed
    startPosition = cursorPosition.getMousePosition();
    selectionAreaTransform.gameObject.SetActive(true);
    isSelecting = true;
}
if (Input.GetMouseButton(0) && isSelecting == true)
{
    // create field to select units
    Vector2 currentMousePosition = cursorPosition.getMousePosition();
    Vector2 lowerLeft = new Vector2(
        Mathf.Min(startPosition.x, currentMousePosition.x),
        Mathf.Min(startPosition.y, currentMousePosition.y)
        );
    Vector2 upperRight = new Vector2(
        Mathf.Max(startPosition.x, currentMousePosition.x),
        Mathf.Max(startPosition.y, currentMousePosition.y)
        );
    selectionAreaTransform.position = lowerLeft;
    selectionAreaTransform.localScale = upperRight - lowerLeft;
}
if (Input.GetMouseButtonUp(0) && isSelecting == true)
{
    // get curosor cordinates, when LPM state is changed
    selectionAreaTransform.gameObject.SetActive(false);
    Collider2D[] collider2DArray = Physics2D.OverlapAreaAll(startPosition, cursorPosition.getMousePosition());
```

Rysunek 15 fragment kodu odpowiedzialnego za zaznaczanie myszą komputerową

5.3. Budowanie

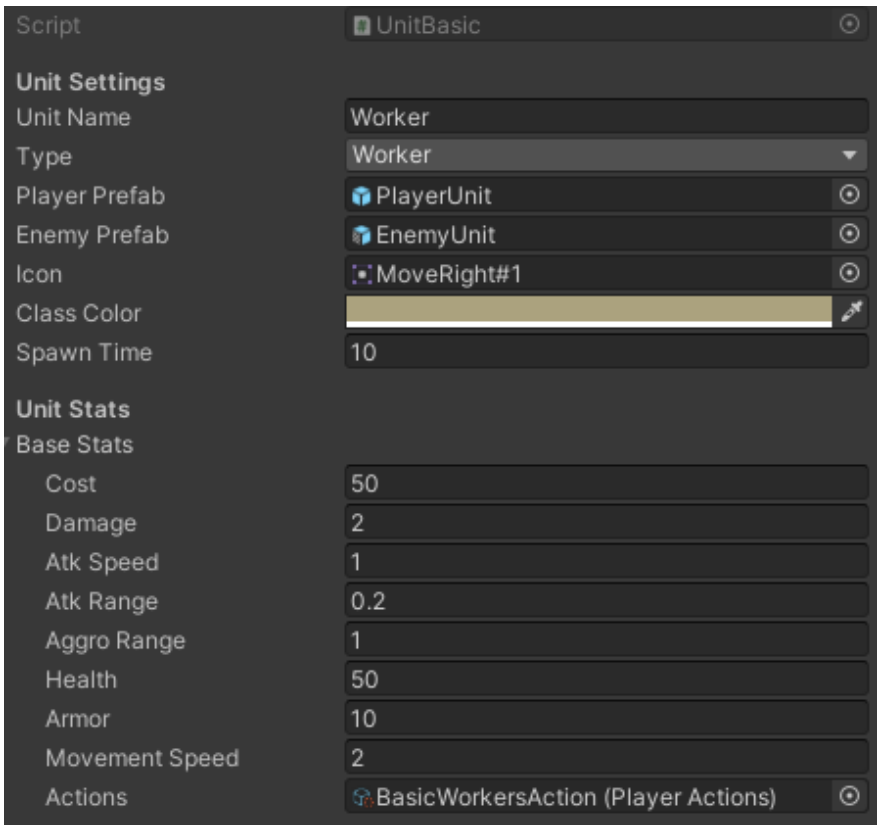
Gracz tworzy struktury przy pomocy jednostek robotników. Zaznaczenie robotnika skutkuje wyświetleniem w ramce akcji możliwych opcji budowania. Po naciśnięci przycisku, w miejscu kursora pojawia się schemat, czyli, przezroczysta grafika przedstawiająca wybraną strukturę, która do momentu podjęcia kolejnej czynności będzie śledziła pozycję kursora. Kliknięcie lewego przycisku myszy powoduje postawienie struktury, jeżeli nie występuje kolizja z już istniejącymi obiektami na mapie. Użycie prawego przycisku powoduje anulowanie budowania i usunięcie schematu.

```
if (scheme != null)
{
    scheme.transform.position = cursorPosition.getMousePosition();
    isFreeSpace = CheckIfFreeSpace(
        cursorPosition.getMousePosition(),
        new Vector2(
            scheme.GetComponent<SpriteRenderer>().bounds.size.x,
            scheme.GetComponent<SpriteRenderer>().bounds.size.y
            ));
    if (isFreeSpace)
    {
        schemeColor.a = 0.75f;
        scheme.GetComponent<SpriteRenderer>().color = schemeColor;
    }
    else scheme.GetComponent<SpriteRenderer>().color = new Color(128f, 0, 0, 0.75f);
}
if (isHoldingAScheme && Input.GetMouseButtonDown(0) && isFreeSpace)
{
    SpawnNewBuilding(camera.ScreenToWorldPoint(Input.mousePosition), buildingType.name);
    isHoldingAScheme = false;
    Destroy(scheme.gameObject);
}
if (isHoldingAScheme && Input.GetMouseButtonDown(1))
{
    isHoldingAScheme = false;
    Destroy(scheme.gameObject);
}
```

Rysunek 16 skryptu odpowiedzialnego za budowanie

5.4. Jednostki

Wszystkie występujące w grze jednostki mają zestaw obiektów, odpowiedzialnych za wizualną część obiektu oraz podstawowe statystyki odpowiedzialne za mechanikę, czyli koszt, punkty zadawanych obrażeń, prędkość ataku, zasięg ataku, zasięg agresji, punkt życia, pancerz, prędkość poruszania się oraz opcjonalny zestaw akcji do wykonania, zawierający dostępne struktury do zbudowania przez daną jednostkę.



Rysunek 17 zestaw statystyk oraz obiektów przypisanych dla jednostki pracownika

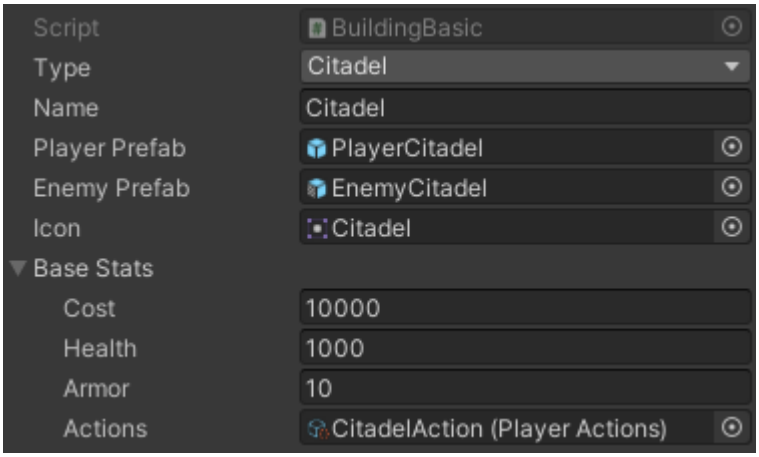
Dane o każdej jednostce są przechowywane w kontenerze danych „ScriptableObject” co pozwala zmniejszyć zużycie pamięci, poprzez unikanie tworzenia dużej ilości prefabrykatów.

Jednostki dodane do sceny umieszczane są w specjalnej strukturze, obiekcie przeznaczonym wyłącznie dla danego typu jednostki, który z kolei znajduje się w jednym z dwóch głównych obiektów nazwanych „PlayerUnits” oraz „EnemyUnits”, odpowiednio dla gracza oraz przeciwnika.

Rysunek 18 struktura przechowywania obiektów

5.5. Budynki

Występujące w grze typy budynków podobnie jak jednostki posiadają przypisane obiekty oraz statystki. W odróżnieniu od jednostek, struktury mają mniej wartości a pole akcji przechowuje informacje, które jednostki struktura może zrekrutować.



Rysunek 19 zestaw statystyk oraz obiektów przypisanych do budynku cytadela

Rysunek 20 struktura przechowywania budynków

Sposób implementacji budynków różni się od jednostek, tym, że prefabrykaty zawierające graficzne elementy dodatkowo zawierają skrypty odpowiedzialne za funkcjonalność danej struktury, zwiększanie ilość złota właściciela budynku oraz skrypt obsługujący rekrutację jednostek.

Skrypt odpowiedzialny za rekrutację może przyjąć listę obiektów do rekrutacji, które kolejno w odpowiednim czasie pojawią się na pozycji pod budynkiem, gdzie wartość pozycji x pozostaje identyczna, natomiast pozycji y jest zmniejszona o połowę wysokości grafiki struktury, a następnie usuwane są z listy. Komponent będzie się powtarzał do momentu aż lista będzie pusta.

```
public void Spawn()
{
    string objectName = spawnTypes[0].ToString() + "s";
    GameObject unit = Instantiate(
        spawnQueue[0],
        new Vector3(
            transform.position.x,
            transform.position.y - gameObject.transform.Find("Sprite").GetComponent<SpriteRenderer>().bounds.size.y/2,
            transform.position.z
            ),
        Quaternion.identity,
        objectToStoreUnits.Find(objectName.Replace(" ", ""))
        );
    objectName = objectName.Substring(0, objectName.Length - 1).ToLower();

    Units.UnitRTS sideUnit = unit.GetComponent<Units.UnitRTS>();

    Units.UnitBasic settings = Units.UnitHandler.instance.GetUnitSettings(objectName);
    sideUnit.baseStats = settings.baseStats;
    sideUnit.gameObject.transform.GetChild(0).GetComponent<SpriteRenderer>().color = settings.classColor;
    sideUnit.GetComponent<Pathfinding.AIPath>().maxSpeed = settings.baseStats.movementSpeed;
    data.unitsRecruted++;
}
```

Rysunek 21 skrypt odpowiedzialny za stworzenie jednostki

Obsługa pasywnego zarabiania budynków bazuje na okresowym dodawaniu złota po czasie podanym w sekundach. Wartości są zawarte w komponentach obiektu. Struktury w grze odpowiedzialne między innymi za zwiększanie zarobków to:

- Cytadela
- Bank
- Dom
- Farma

```
public IEnumerator GetMoneyPassive(int gold)
{
    yield return new WaitForSeconds(secondsToGold);
    statistics.GetMoney(gold);
    StartCoroutine(GetMoneyPassive(gold));
}
```

Rysunek 22 funkcja używana do okresowego dodawania złota

5.6. Walka

Walka następuje, kiedy jednostka atakuje obiekt nieprzyjaciela, nie musi w tym samym czasie, być celem ataku. Celem może zostać jednostka jak również budynek.

```
internal void CheckForEnemyTargets(float aggroRange)
{
    rangeColliders = Physics2D.OverlapCircleAll(transform.position, aggroRange);

    Transform aggroTmp = null;
    for (int i = 0; i < rangeColliders.Length; i++)
    {
        if(aggroTmp==null &&
            rangeColliders[i].gameObject.layer != gameObject.layer &&
            rangeColliders[i].gameObject.layer != gameObject.layer + 1 &&
            rangeColliders[i].gameObject.layer != 7 //7 layer is world obstacles
            )
        {
            aggroTmp = rangeColliders[i].gameObject.transform;
        }
        else if (rangeColliders[i].gameObject.layer != gameObject.layer &&
            rangeColliders[i].gameObject.layer != gameObject.layer + 1 &&
            rangeColliders[i].gameObject.layer != 7 &&
            Vector2.Distance(aggroTmp.transform.position,gameObject.transform.position)>
            Vector2.Distance(rangeColliders[i].gameObject.transform.position, gameObject.transform.position)
            )
        {
            aggroTmp = rangeColliders[i].gameObject.transform;
        }
    }
    aggroTarget = aggroTmp;
    if (aggroTarget != null) hasAggro = true;
}
```

Rysunek 23 funkcja odpowiedzialna za poszukiwanie wrogiej jednostki

Jednostki posiadają funkcje za pomocą, której szukają przeciwnika w zasięgu ustalonym w specyfikacji jednostki, jeżeli przeciwnik znajdzie się w promieniu zasięgu agresji, jednostka rozpocznie pościg, który może doprowadzić do dwóch sytuacji:

- Porzucenie – następuje w momencie, kiedy dystans między agresorem a celem będzie większy od dwukrotności promienia zasięgu agresji.
- Atak – agresor zbliża się do celu, jeżeli znajdzie się w zasięgu ataku zadaje obrażenia celowi, następnie kontynuuje pościg.

Gracz posiada możliwość nadania komendy ataku jednostce, w takiej sytuacji jednostka kontynuuje atak do momentu otrzymania innej komendy np. ruchu. W przypadku nadania komendy, jednostka nie może porzucić ścigania przeciwnika.

Doprowadzenie do utraty wszystkich punktów życia przez cel powoduje zniszczenie jednostki, a agresor ponownie przechodzi do szukania nowych przeciwników w obrębie promienia agresji.

Wzór na podstawie, którego są wyliczane obrażenia:

$$f(obrazenia) = \begin{cases} obrazenia - pancerz & dla \quad obrazenia + 1 > pancerz \\ 1 & dla \quad obrazenia + 1 <= pancerz \end{cases}$$

5.7. Szukanie ścieżki

W grze występują obiekty, które będą stanowiły przeszkodę dla przemieszczających się jednostek, przykład stanowią struktury. W celu uniknięcia niepożądanego zatrzymywania się jednostek w miejscu, użyty został system A* Pathfinding Project[10]. System został użyty ze względu na obsługę wyszukiwania ścieżek w projektach 2D.

System bazuje na algorytmie wyszukiwania A* działającego na grafach, system obsługując aplikację 2D używa kafelkowej mapy, w której kafelki pełnią rolę wierzchołków grafu, z kolei połączenia między kafelkami są krawędziami[11].

Algorytm A* zaczyna od pierwszego węzła i uwzględnia wszystkie sąsiednie wierzchołki, które są niedostępne i stanowią przeszkodę, są odfiltrowane, następnie algorytm podejmuje decyzje o wybraniu wierzchołka o najniższej wartości ruchu od punktu startowego. Wartość ruchu składa się z sumy kosztu przejścia z komórki początkowej do komórki bieżącej oraz wartości heurystycznej, czyli oszacowanemu kosztowi przejścia z bieżącej komórki do ostatniej. Algorytm powtarzany jest rekurencyjnie do uzyskania najkrótszej ścieżki do punktu docelowego[12].

Do oszacowania wartości heurystycznej używa się heurystyki, odpowiedniej to sposobu poruszania się obiektu.

The Manhattan Distance (metryka miasto) – używana jest, gdy obiekt ma się poruszać wyłącznie w czterech kierunkach.

The Euclidean Distance (metryka euklidesowa) – stosowana jest w sytuacji, kiedy kierunek, w którym obiekt się porusza nie jest ograniczony[13].

[10] https://arongranberg.com/astar/
[11] http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html
[12] https://www.educative.io/edpresso/what-is-the-a-star-algorithm
[13] https://brilliant.org/wiki/a-star-search/

5.8. Strategia przeciwnika

Przeciwnik zawiera określone funkcje, odpowiadające za ruch jednostek oraz rozbudowę bazy. Program w sposób losowy wybiera jaką czynność ma podjąć.

5.8.1. Jednostki

Jednostki przeciwnika są sterowane za pomocą skryptu, który wydaje komendy, mające na celu atak na bazę gracza. Skrypt wybiera losową ilość jednostek przeciwnika dostępnych na mapie, których liczba stanowi od 10% do 50% wszystkich.

Wybrany oddział jest przemieszczany do wyznaczonego punktu zbiórki skąd po określonym czasie grupa rusza w kierunku struktur gracza. Priorytetem ataku jest główny budynek gracza, jeżeli został zniszczony wybierana jest inna struktura gracza na mapie, a następnie tam wysyłane są kolejne ataki. Zapobiega to sytuacji, w której gracz buduje struktury daleko do swojego głównego budynku.

Jednostki po otrzymaniu komendy zostają usunięte z listy dostępnych jednostek, jednocześnie będąc oznaczone jako wykonujące komendę, dzięki czemu nie zostaną one wybrane ponownie.

W sytuacji, kiedy jednostka zostanie bezczynna po wykonaniu komendy, zostaje ona zawrócona do punktu zbiórki oraz przydzielona do listy dostępnych jednostek.

```
public IEnumerator CheckIfReturningToBase()
{
    var positionTmp = gameObject.transform.position;
    yield return new WaitForSeconds(5);
    if (positionTmp == gameObject.transform.position)
    {
        IfCommand = false;
        MoveTo(GameObject.Find("EnemyAI").GetComponent<EnemyUnitAI>().groupPoint);
    }
}
```

Rysunek 24 funkcja odpowiedzialna za powrót jednostki

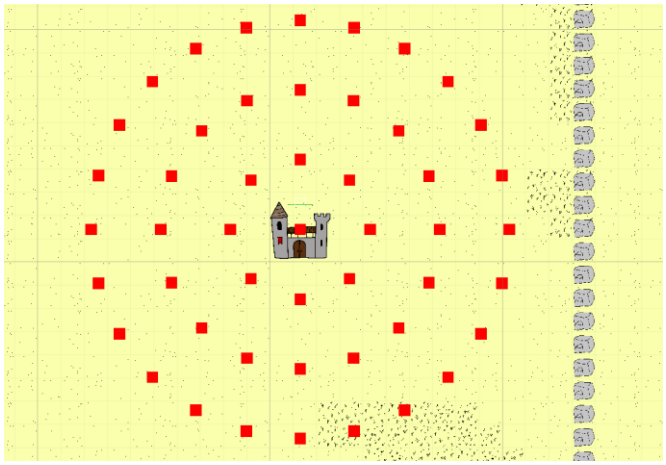
5.8.2. Budynki

Skrypt odpowiedzialny za budynki zawiera funkcje obsługujące budowę struktur jak również rekrutację jednostek. Wymienione funkcję są wykonywane w sytuacji, kiedy przypisane przeciwnikowi złoto osiągnie odpowiednią wartość.

Złoto przeciwnika jest przydzielane do dwóch zmiennych, część wszystkich środków jest przydzielona na cel rekrutacji jednostek, druga część do budowy struktur.

Rekrutacja jednostek jest wykonywana w losowym budynku przeciwnika z dostępnych na mapie. Z listy dostępnych dla danego typu budynku losowana jest jednostka, która zostaje wyznaczona do rekrutacji. Skrypt będzie czekał do momentu, aż przydzielone złoto osiągnie wymaganą wartość, po czym zleci rekrutację.

Budowa struktur jest realizowana, jeżeli odpowiednia ilość złota jest przydzielona oraz istnieje wolne miejsce na dostępnej liście wygenerowanych pozycji.



Rysunek 25 wizualizacja generowanych pozycji

Algorytm budowy struktur w pierwszej kolejności wybierze budynek o nazwie Cytadela, jeżeli został zniszczony oraz pozostałe warunki budowy są spełnione. W przeciwnym wypadku, jeżeli Cytadela nie została zniszczona, skrypt wybierze losową strukturę z listy dostępnych budynków, jeżeli odpowiednie warunki są spełnione, istnieją robotnicy oraz przeciwnik ma uzbieraną odpowiednią ilość złota.

5.9. Warunki zwycięstwa

Celem gry jest zniszczenie budynków jednej ze stron. Komponent aktualizuje liczbę znajdujących się na mapie budynków, jeżeli w trakcie rozgrywki dojdzie do utraty budynków jednej ze stron gra zakończy się i zostanie wyświetlony element interfejsu wyświetlający skrócone podsumowanie rozgrywki.

```
void Update()
{
    UpdatePlayerBuildings();
    if (playerBuildingsCount == 0 && !ifEndOfGame)
    {
        DefeatScreen();
        ifEndOfGame = true;
    }
    UpdateEnemyBuildings();
    if (enemyBuildingsCount == 0 && !ifEndOfGame)
    {
        VictoryScreen();
        ifEndOfGame = true;
    }
}
```

Rysunek 26 fragment skryptu odpowiadającego za warunki zwycięstwa

Rysunek 27 ekran po osiągnięciu zwycięstwa w grze

5.10. Statystki

Zaimplementowana jest wizualizacja przebiegu gry po zakończeniu, przy pomocy danych zbieranych w trakcie. W każdej sekundzie gry dodawany jest rekord do listy zawierając wartości o aktualnym stanie złota, jednostek oraz w jakim czasie dany rekord został dodany. Wszelkie dane stanowiące podsumowanie rozgrywki, takie jak całkowita ilość uzbieranego złota czy jednostek jest na bieżąco aktualizowana.

```
public class DataRecord
{
    public float time;
    public float money;
    public int units;
```

Rysunek 28 klasa zawierająca wartości wpisywane okresowo

```
public List<DataRecord> datas = new List<DataRecord>();

public float moneyCollected;
public int unitsRecruted = 0;
public float timer = 0.0f;
public bool isVictory;
```

Rysunek 29 zmienne zawarte w komponencie przechowującym statystyki

Na podstawie zebranych danych generowane są wykresy, zależnie od wybranej opcji, dostępne wykresy to:

- Ilość jednostek o danym czasie
- Ilość złota o danym czasie
- Ilość punktów o danym czasie

Rysunek 4 ekran po zakończonej rozgrywce z wyświetlonym wykresem zależności między złotem a czasem rozgrywki

Funkcja zaczyna od wygenerowania wykresu, poprzez dodanie punktów, których współrzędne stanowią zebrane dane. Sąsiednie punkty połączone są linią prostą.

```
private void ShowGraph(Dictionary<float,float> values)
{
    float graphHeight = graphContainer.sizeDelta.y;
    float graphWidth = graphContainer.sizeDelta.x;
    float xSize = graphWidth/values.Count;
    float yMaximum = 0;
    foreach(KeyValuePair<float, float> datas in values)
    {
        if(datas.Value>yMaximum) yMaximum = datas.Value;
    }
    yMaximum = MaximumY(yMaximum);

    int i = 0;
    GameObject lastPoint = null;
    foreach(KeyValuePair<float, float> datas in values)
    {
        float xPosition =  i * xSize;
        float yPosition = (datas.Value / yMaximum) * graphHeight;
        GameObject point = AddPoint(new Vector2(xPosition, yPosition));
        if(lastPoint != null)
            CreateLines(lastPoint.GetComponent<RectTransform>().anchoredPosition,
                point.GetComponent<RectTransform>().anchoredPosition);
        lastPoint = point;

        i++;
    }
```

Rysunek 30 fragment funkcji generującej wykres

Kolejnym krokiem jest wygenerowanie odpowiednich przedziałów na osiach. Przedziały na osiach, stanowią 0%, 10%, … i 100% wartości maksymalnej. Oś y, stanowi przedstawienie ilości jednostek, złota lub punktów, aby wartości prezentowały się korzystniej została wprowadzona funkcja, która zaokrągla maksymalną wartość na osi y w górę, uwzględniając dwie pierwsze cyfry wartości maksymalnej odpowiednio je zwiększając. Dzięki temu zabiegowi wartości na wykresie nie będą docierały do górnej granicy okna oraz przedziały będą korzystniej się prezentowały.

```
public float MaximumY(float maxValue)
{
    int length = maxValue.ToString().Length;
    float largestDigit = maxValue/Mathf.Pow(10, length-1);
    float rest = maxValue%Mathf.Pow(10, length-1);
    maxValue = (int)largestDigit * Mathf.Pow(10, length-1);

    largestDigit = rest/Mathf.Pow(10, length-2);
    if(largestDigit >= 0) largestDigit = (int)largestDigit + 1;
    maxValue += largestDigit * Mathf.Pow(10, length-2);

    return maxValue;
}
```

Rysunek 31 funkcja zaokrąglająca wartość w górę

5.11. Komendy

Do gry w celu ułatwienia testowania funkcji użytych w grze została zaimplementowana konsolka oraz komendy wywołujące pewne zachowania, które w normalnej rozgrywce nie są pożądane.

W celu osiągnięcia zezwolenia na używanie komend, należy wpisać następujący ciąg znaków na scenie głównej: „cheat”. Kolejnym krokiem jest kliknięcie klawisza „ ` ” (grawis), spowoduje to wyświetlenie konsolki, w której wpisanie komendy spowoduje odpowiedni efekt.

Zaimplementowane komendy:

- Help – wyświetla dostępne komendy.
- Win – powoduje usunięcie budynków przeciwnika, którego efektem jest wygrana.
- Lose - powoduje usunięcie budynków gracza, którego efektem jest przegrana.
- Gold – dodaje do konta gracza 10 000 złota.

```
if(!cheatsAvailable)
{
    foreach(KeyCode vKey in System.Enum.GetValues(typeof(KeyCode)))
    {
        if(Input.GetMouseButtonDown(0) || Input.GetMouseButtonDown(1) || Input.GetMouseButtonDown(2))
        {}
        else if(Input.GetKeyDown(vKey))
        {
            if(vKey.ToString()==keyCode) keyCode = null;
            else keyCode = vKey.ToString();
        }
    }
    if(keyCode == "C")
    {
        pressed = keyCode;
        keyCode = null;
        getKeys = true;
    }
    if(getKeys && keyCode!=null)
    {
        pressed += keyCode;
        keyCode = null;
    }

    if(pressed == "CHEAT") cheatsAvailable = true;
    else if(pressed.Length>6) getKeys =false;
}
else
{
    if(Input.GetKeyDown(KeyCode.BackQuote) && cheatsAvailable)
    {
        if(panel.active == true) panel.SetActive(false);
        else if(panel.active == false) panel.SetActive(true);
    }
```

Rysunek 32 fragment skryptu odpowiedzialnego za aktywację konsoli

5.12. Grafika

Grafiki wszystkich obiektów zostały stworzone za pośrednictwem programu GIMP. Jednostkami są uzbrojone „ziemniaki”, animacja przedstawiająca ich ruch oraz nadająca życia obiektom, została stworzona na podstawie klatek będących pojedynczymi grafikami.

Struktury przedstawiają budynki inspirowane przykładowymi tytułami oraz średniowiecznymi motywami architektury.

Czcionka Bubble font[14] została zaczerpnięta z Unity Asset Store, jej twórcą jest Jazz Create Games. Użyta, czcionka nadaje grze bajkowego uroku. Czcionka nadaje dodatkowy efekt wizualny, jednocześnie pozostając przejrzysta.

[14] https://assetstore.unity.com/packages/2d/fonts/bubble-font-free-version-24987

6. Zakończenie

Wynikiem przeprowadzonej pracy jest gra strategiczna czasu rzeczywistego, gra zawiera niezbędne funkcjonalności do prawidłowego działania. Gra zawiera znaczącą większość założonych funkcjonalności. Architektura gry pozwala na dalsze ulepszenia czy zmiany w elementach gry, które ją urozmaicą.

W przyszłości planowane jest wprowadzanie sugestii jakie powstaną w wyniku testowania przez grupę odbiorców, naprawianie ewentualnych błędów oraz ulepszenie skryptów odpowiedzialnych za decyzje przeciwnika.

7. Bibliografia

[1] https://www.britannica.com/topic/StarCraft
[2] http://videogamesandbooze.blogspot.com/2010/
[3] https://allegro.pl/artykul/starcraft-2-battle-chest-recenzja-gry-LvKnem1VLia
[4] https://www.ign.com/articles/2016/03/22/the-rise-and-fall-of-starcraft-ii-as-an-esport
[5] https://en.wikipedia.org/wiki/The_Lord_of_the_Rings:_The_Battle_for_Middle-earth
[6] https://www.moddb.com/mods/bfme-patch-108/images/rohan
[7] https://www.statista.com/statistics/268237/global-market-share-held-by-operating-systems-since-2009/
[8] https://en.wikipedia.org/wiki/Unity_(game_engine)
[9] https://www.gimp.org/about/
[10] https://arongranberg.com/astar/
[11] http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html
[12] https://www.educative.io/edpresso/what-is-the-a-star-algorithm
[13] https://brilliant.org/wiki/a-star-search/
[14] https://assetstore.unity.com/packages/2d/fonts/bubble-font-free-version-24987
[15] https://docs.unity3d.com/Manual/index.html
[16] https://docs.microsoft.com/pl-pl/dotnet/csharp/
[17] https://www.youtube.com/c/Brackeys
[18] https://www.youtube.com/c/CodeMonkeyUnity
[19] https://arongranberg.com/astar/docs/
[20] https://www.chosic.com/download-audio/28027/