

Maciej Bronikowski 248838, Paweł Nita 248847

ETAP 3 - (F21) Trudności i złożoność projektu

Diagram klas



Opis wymagań funkcjonalnych projektowanego systemu:

Użytkownik:

Nazwa funkcji: Wybór pozycji w menu

Opis: W oknie aplikacji przedstawione jest menu, zawierające opcje: rozpoczęcia i zakończenia programu, użytkownik wybiera daną opcję

Nazwa funkcji: Rozpoczęcie symulacji

Opis: Użytkownik wybiera z menu opcję rozpoczęcia, startuje główna symulacja

Nazwa funkcji: Wybór edycji mapy

Opis: Użytkownik wybiera z menu opcję rozpoczęcia, uruchamia się opcja edycji mapy

Nazwa funkcji: Edycja mapy

Opis: Użytkownik wybiera typ podłoża oraz jego pozycję za pomocą myszki i klawiatury

Nazwa funkcji: Zakończenie symulacji

Opis: Użytkownik wybiera z menu opcję zakończenia programu, zamykane jest okno aplikacji

Nazwa funkcji: Przesuwanie widoku symulacji

Opis: Podczas pracy symulacji, użytkownik ma możliwość zmiany pozycji kamery, aby śledzić symulację z wybranego miejsca

Nazwa funkcji: Przeglądanie statystyk symulacji

Opis: Za pomocą interfejsu graficznego aplikacji, użytkownik ma możliwość śledzenia postępów symulacji, najważniejszych danych typu ilość zebranych surowców, populacja

System symulacji:

Nazwa funkcji: Przeglądanie danych symulacji

Opis: System zbiera dane na temat konkretnych postaci oraz ogólnego postępu symulacji

Nazwa funkcji: Sprawdzanie dostępnych surowców

Opis: Sprawdzana jest ilość posiadanych surowców przez grupę

Nazwa funkcji: Sprawdzanie obecnego zadania postaci

Opis: Sprawdzane jest obecnie wykonywane zadanie postaci, jeżeli postać nie ma określonego zadania zostaje jej nadane zadanie, które ma przybliżyć ukończenie symulacji

Nazwa funkcji: Sprawdzanie pozycji

Opis: Sprawdzana jest obecna pozycja na mapie postaci na podstawie której nadawane są kolejne zadania

Nazwa funkcji: Wybieranie kolejnego zadania postaci

Opis: Nadawane jest nowe zadanie dla postaci, która w obecnej chwili nie wykonuje pracy

Nazwa funkcji: Tworzenie budynku

Opis: Jedno z zadań, które może wykonywać postać, nadawane jest gdy posiadana jest określona ilość surowców i wymagana jest większa ilość budynków

Nazwa funkcji: Określanie optymalnego miejsca dla budynku

Opis: Wyliczane jest optymalne położenie nowopowstałego budynku, na podstawie pobranych pozycji postaci

Nazwa funkcji: Zbieranie surowców

Opis: Zadanie przydzielane wolnym postaciom, w celu zwiększenia ilości posiadanych surowców

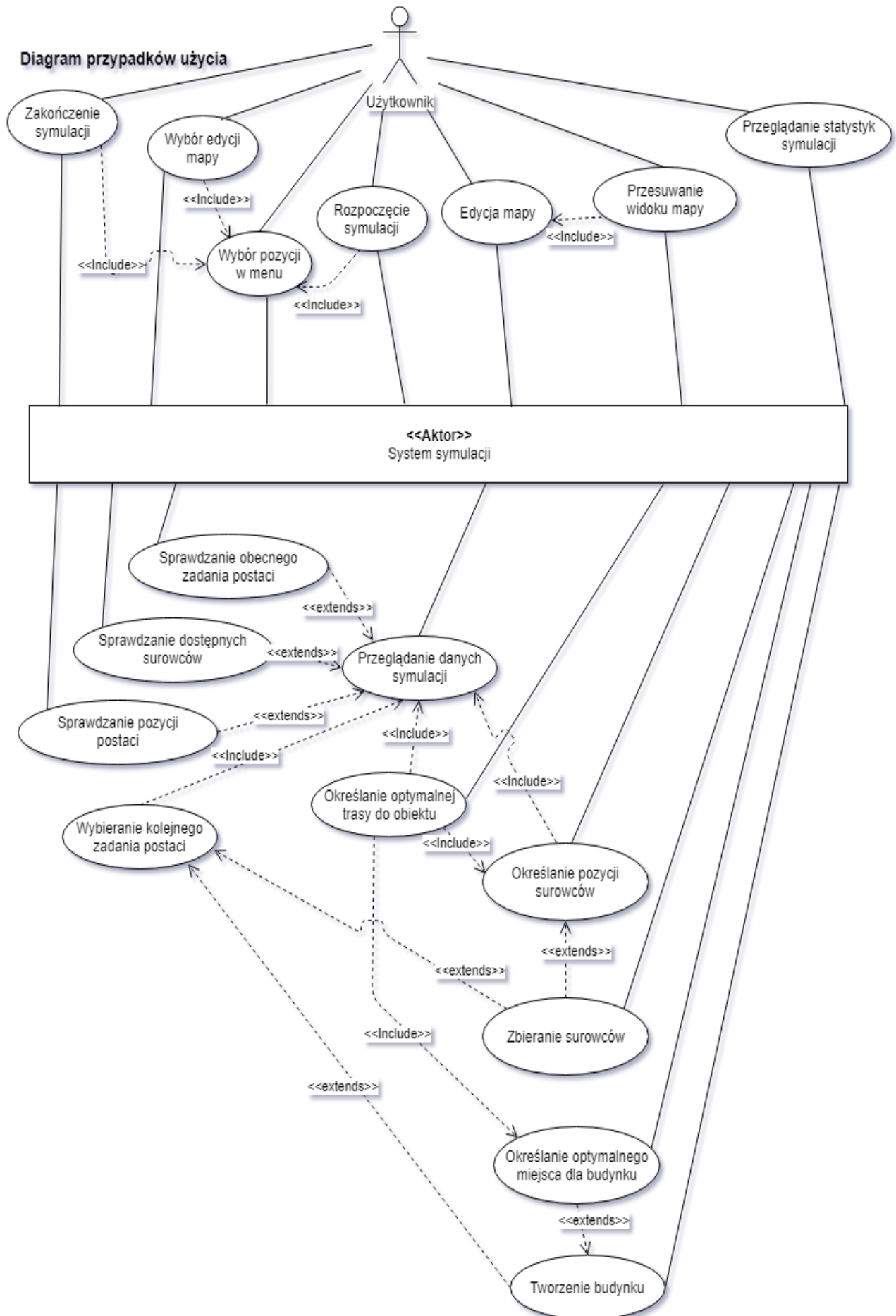
Nazwa funkcji: Określanie pozycji surowców

Opis: Sprawdzana jest obecna pozycja surowców, w celu porównania jej z pozycją postaci

Nazwa funkcji: Określenie optymalnej trasy do obiektu

Opis: Na podstawie pozycji postaci oraz obiektów na których możliwa jest wykonana akcja, wybierany jest odpowiedni obiekt, zależny od wybranego zadanie i obliczana jest optymalna trasa

Diagram przypadków użycia



Analiza czasownikowo-rzeczownikowa

Do działania **symulacji** potrzebny obiekt główny **zawierający w sobie pętlę główną**, **ładowane w nim będą do pamięci informacje** o **czcionce** i **teksturach** używanych w aplikacji. Cała aplikacja podzielona będzie na **stany**: **menu**, **edytor mapy** i **czas właściwej symulacji**. Główna symulacja będzie **uruchamiana z menu** za pomocą **wyboru** określonego, animowanego **przycisku**, przy użyciu myszki lub klawiatury. W menu będzie również możliwość **zamknięcia aplikacji**, zanim uruchomiona zostanie symulacja. Gdy wybrana zostanie opcja edycji mapy, użytkownik będzie miał możliwość **umieszczania wybranych typów podłoża na mapie**, na której będzie odbywała się symulacja. Dane o mapie będą przechowywane w **pliku binarnym**, obsługiwane za pomocą **specjalnej klasy**. Po symulacji uruchomieniu **narysowana** będzie interaktywna **mapa** o określonej wielkości, z możliwością **przesunięcia** za pomocą myszy. Na mapie **rysowane** będą **obiekty interaktywne** i **postacie**, **wykonujące zadania** przybliżające je do osiągnięcia celu. **Wyświetlane** będą tylko obiekty znajdujące się w **polu widzenia kamery**. Mapa będzie składała się z **kafelków** o określonej wielkości i położeniu. Wszystkie **tekstury** oraz **sprite'y** będą **przechowywane** w osobnych klasie oraz **przekazywane** jako wskaźniki. Tekstury będą składały się z kafelków, na podstawie których **wyświetlane** będą **animacje**, informacje o nich **zawierać** będzie oddzielna klasa. Samą animacją i **informacjami o danym stanie animacji** **zajmować** się będzie klasa specjalnie do tego przeznaczona. Wszystkie obiekty na mapie będą podzielone na **postacie** i obiekty na których postacie będą mogły wykonać określoną akcję. Każdy rodzaj obiektu będzie miał swój własny typ i charakterystykę, zostaną one podzielone na obiekty z których postacie będą mogły **wydobywać** określone **surowce** oraz na **budynki**, które zostaną **stworzone** za pośrednictwem postaci, każdy budynek będzie charakteryzował się określoną **listą korzyści** płynących z jego posiadania, przykładowo w pobliżu **ogniska** postacie **regenerować** będą swoje **zdrowie**, a w **magazynie** **przechowywane** będą **surowce**, każdy budynek **posiada swój koszt**. **Miejsce postawienia budynku** będzie **ustalane** na podstawie pozycji innych obiektów. Postacie będą **oddziaływać** na obiekty surowców, aby osiągnąć cel symulacji, tutaj określony jako zebranie danej ilości pożywienia lub drewna. Każda postać będzie **wykonywać określone zadanie**, o najwyższym priorytecie. Każdy ruch postaci będzie **obliczany**, **ścieżka** będzie zależna od **odległości** danego typu obiektu i przeszkód.

Karty CRC

Wszystkie klasy z namespace sf są klasami biblioteki sfml (biblioteki graficznej).

Simulation	
Klasy bazowe: brak	
Klasy potomne: brak	
Odpowiedzialności	Współpracownicy
<ul style="list-style-type: none">✓ Obsługa pętli głównej symulacji✓ Ładowanie do pamięci tekstur i czcionek✓ Przejsie między widokiem menu i widokiem symulacji	SimulationState SimWindow TextureMenager SpriteDividedMenager sf::Sprite sf::Font

SimulationState	
Klasy bazowe: brak	
Klasy potomne: SimulationStateMenu, SimulationStateEditor	
Odpowiedzialności	Współpracownicy
<ul style="list-style-type: none">✓ Obsługa danych wejściowych jak klawiatura i mysz✓ Aktualizacja obiektów z uwzględnieniem czasu, który minął między poprzednią klatką, a obecną✓ Wyświetlanie obiektów na ekranie	Simulation

SimulationStateMenu	
Klasy bazowe: SimulationState	
Klasy potomne: brak	
Odpowiedzialności	Współpracownicy
<ul style="list-style-type: none">✓ Obsługa danych wejściowych jak klawiatura i mysz✓ Wyświetlanie obrazu ekranu głównego i menu✓ Aktualizacja obiektów z uwzględnieniem czasu, który minął między poprzednią klatką, a obecną✓ Ładowanie symulacji, lub zamknięcie okna	Button SimWindow sf::View sf::Sprite

Button	
Klasy bazowe: brak	
Klasy potomne: brak	
Odpowiedzialności	Współpracownicy
<ul style="list-style-type: none">✓ Obsługa wyświetlania przycisku✓ Obsługa animacji z uwzględnieniem czasu, który minął między poprzednią klatką, a obecną✓ Sprawdzanie, czy pozycja myszy pokrywa się z pozycją przycisku	SimWindow sf::Text sf::Vector2f

SimulationStateMain	
Klasy bazowe: SimulationState	
Klasy potomne: brak	
Odpowiedzialności	Współpracownicy
<ul style="list-style-type: none"> ✓ Obsługa danych wejściowych jak klawiatura i mysz ✓ Aktualizacja obiektów z uwzględnieniem czasu, który minął między poprzednią klatką, a obecną ✓ Wyświetlanie obiektów na ekranie 	SimView Map Man StaticObject PopulationAi sf::View

SimulationStateEditor	
Klasy bazowe: SimulationState	
Klasy potomne: brak	
Odpowiedzialności	Współpracownicy
<ul style="list-style-type: none"> ✓ Obsługa danych wejściowych jak klawiatura i mysz ✓ Wyświetlanie na bieżąco modyfikowanej mapy 	SimView Map sf::View BinaryFileManager

SimView	
Klasy bazowe: sf::View	
Klasy potomne: brak	
Odpowiedzialności	Współpracownicy
<ul style="list-style-type: none"> ✓ Podstawowa funkcjonalność sf::View ✓ Przewijanie widoku po otrzymaniu danych o pozycji kursora po spełnieniu odpowiednich wymagań 	

Map	
Klasy bazowe: brak	
Klasy potomne: brak	
Odpowiedzialności	Współpracownicy
<ul style="list-style-type: none"> ✓ Przechowywanie wielkości mapy ✓ Obsługa wyświetlania segmentów mapy tylko w miejscu wskazywanym przez widok 	Tile std::vector sf::Vector BinaryFileManager

Tile	
Klasy bazowe: brak	
Klasy potomne: brak	
Odpowiedzialności	Współpracownicy
<ul style="list-style-type: none"> ✓ Przechowywanie informacji o rodzaju klocka i miejscu odpowiedniej tekstury ✓ Dopasowywanie podczas rysowania tekstury do sprite i rysowanie go w odpowiednim miejscu mapy 	

SimWindow	
Klasy bazowe: sf::RenderWindow	
Klasy potomne: brak	
Odpowiedzialności	Współpracownicy
<ul style="list-style-type: none"> ✓ Podstawowa funkcjonalność klasy sf::RenderWindow ✓ Ładowanie odpowiedniej konfiguracji okna 	sf::Vector2i sf::String

TextureMenager	
Klasy bazowe: brak	
Klasy potomne: brak	
Odpowiedzialności	Współpracownicy
<ul style="list-style-type: none"> ✓ Przechowywanie listy tekstur ✓ Zwracanie wskaźnika wybranej tekstury 	std::map sf::Texture

SpriteDividedMenager	
Klasy bazowe: brak	
Klasy potomne: brak	
Odpowiedzialności	Współpracownicy
<ul style="list-style-type: none"> ✓ Przechowywanie listy sprite'ów ✓ Zwracanie wskaźnika wybranego Sprite'a 	std::map sf::Sprite SpriteDivided AnimationHandler

SpriteDivided	
Klasy bazowe: sf::Sprite	
Klasy potomne: brak	
Odpowiedzialności	Współpracownicy
<ul style="list-style-type: none"> ✓ Podstawowa funkcjonalność sf::Sprite ✓ Przechowywanie informacji o wielkości części tekstury do wycięcia dla danego sprite ✓ Przechowywanie informacji o ilości części w teksturze 	Sf::Vector2u

AnimationHandler	
Klasy bazowe: brak	
Klasy potomne: brak	
Odpowiedzialności	Współpracownicy
<ul style="list-style-type: none"> ✓ Przejście między klatkami po upływie odpowiedniej ilości czasu ✓ Przechowywanie informacji o aktualnej klatce animacji 	

StaticObject	
Klasy bazowe: brak	
Klasy potomne: StaticObjectResouces, StaticObjectHome	
Odpowiedzialności	Współpracownicy
<ul style="list-style-type: none"> ✓ Są to wszystkie obiekty na które postacie będą miały wpływ ✓ Przechowywanie informacji o położeniu obiektu ✓ Przechowywanie informacji o wskaźniku na odpowiedni dla niego rodzaj i wycinek tekstury 	AnimationHandler

StaticObjectWarehouse	
Klasy bazowe: StaticObejct	
Klasy potomne: brak	
Odpowiedzialności	Współpracownicy
<ul style="list-style-type: none"> ✓ Przechowywanie surowców 	

StaticObjectResouces	
Klasy bazowe: StaticObejct	
Klasy potomne: brak	
Odpowiedzialności	Współpracownicy
<ul style="list-style-type: none"> ✓ Przechowywanie informacji o rodzaju wydobywanego surowca ✓ Wyświetlanie obiektu 	

StaticObjectHomeFireplace	
Klasy bazowe: StaticObjectHome	
Klasy potomne: brak	
Odpowiedzialności	Współpracownicy
<ul style="list-style-type: none"> ✓ Odzyskiwanie zdrowia gdy jest się w pobliżu ✓ Przechowywanie informacji na temat kosztu budowy ✓ Sprawdzanie położenia innych obiektów względem siebie 	

Man	
Klasy bazowe: brak	
Klasy potomne: brak	
Odpowiedzialności	Współpracownicy
<ul style="list-style-type: none"> ✓ Przechowywanie informacji o położeniu obecnym i zaplanowanym ✓ Przechowywanie informacji o obecnym zadaniu ✓ Obliczanie kolejnego przesunięcia postaci ✓ Animowanie ruchu postaci ✓ Przechowywanie informacji na temat potrzeb ✓ Przechowywanie informacji o obecnej ilości surowców 	AnimationHandler StaticObject ManAI sf::Sprite

ManAi	
Klasy bazowe: brak	
Klasy potomne: brak	
Odpowiedzialności	Współpracownicy
<ul style="list-style-type: none"> ✓ Obliczanie optymalnej drogi do określonego miejsca ✓ Przechowywanie listy kolejnych zmian położenia postaci 	sf::Vector2i

PeopleManager	
Klasy bazowe: brak	
Klasy potomne: brak	
Odpowiedzialności	Współpracownicy
<ul style="list-style-type: none"> ✓ Przechowywanie listy ludzi ✓ Przechowywanie listy surowców ✓ W zależności od aktualnych potrzeb dokonywanie decyzji na temat kolejnego zadania 	Man

BinaryFileManager	
Klasy bazowe: brak	
Klasy potomne: brak	
Odpowiedzialności	Współpracownicy
<ul style="list-style-type: none"> ✓ Przechowywanie mapy ✓ Odczytywanie pliku z mapą ✓ Nadpisywanie pliku z mapą 	Map