Zespół 14

Skład:

Robert Bielas

Krzysztof Burkat

Michał Ćwiertnia

Karolina Mura

Temat projektu: silnik gier z gatunku symulator ekonomiczny

Kryptonim: bgEngine

Kontekst

Dziedziną analizowanego problemu jest rynek rozrywkowy. Rozważane pole działania sprowadza się do tej gałęzi przemysłu elektronicznej zabawy, której głównym paradygmatem jest rozwój silników symulatorów ekonomicznych. Użytkownikami są gracze i twórcy gier, którzy chcą w prosty i wygodny sposób wygenerować podstawowy element rozgrywki, powtarzający się we wszystkich produkcjach gier typu symulator ekonomiczny - czyli podmienić leżące na najniższej warstwie mechaniki zależności pomiędzy logicznymi reprezentacjami posiadanych zasobów, wcześniej tworząc je. Dalej w tym dokumencie, naszych użytkowników będziemy nazywać po prostu "graczami".

Słownik pojęć:

byty - użytkownik może je stworzyć w kreatorze dla danego kontekstu, aby potem móc z nich korzystać w grze, należą do nich budynki, mieszkańcy i zasoby

domena - przestrzeń wraz ze wszystkimi dostępnymi bytami, którą ma zarządzać użytkownik

drzewka technologiczne - hierarchiczna wizualizacja sekwencji możliwych do zdobycia ulepszeń i technologii

gra - symulacja elementów wygenerowanych przez kreator, które mogą zostać wykorzystane do stworzenia pełnowymiarowej gry komputerowej, jest stworzona na podstawie zdefiniowanego przez użytkownika kontekstu

gracz - patrz: użytkownik

HUD - element panelu w module rozgrywki który wyświetla mapę

kreator - umożliwia stworzenie bytów na podstawie preferencji gracza

kontekst - tworzony dla każdego zbioru powiązań (zbioru związanych ze sobą bytów)

mapa - wizualizacja zarządzanej przez użytkownika domeny

moduł - logiczna część aplikacji realizująca określone funkcjonalności

produkt - końcowa aplikacja ułatwiająca użytkownikowi wygenerowanie elementów, z których może powstać gra i ich przetestowanie

program - patrz: produkt

protokół - definiuje sposób tworzenia zasad obowiązujących w grze

rozgrywka - gra uruchomiona przez użytkownika

samouczek - patrz: tutorial

silnik - podstawowy element kreatora i gry, który kontroluje, aby z kreatora korzystać zgodnie z protokołem, a w grze były przestrzegane określone zasady
struktury - patrz: byty
tutorial - rodzaj przewodnika wprowadzającego użytkownika w zasady, wg. których może grać i ogólnie korzystać z produktu
użytkownik - korzysta z produktu w celu stworzenia nowej gry komputerowej - wymyśla zbiór zasad, definiuje je w kreatorze, generuje wynik swojej pracy i testuje go

Opis problemu

Analizowana sytuacja i jej cechy charakterystyczne - wdrożenie nowego narzędzia do tworzenia produktów elektronicznej zabawy, koncentracja nad właściwościami i wymaganiami problemu, chwilowo bez wyróżniania sposobów jego rozwiązania.

Wstęp:

W dzisiejszych czasach całkiem powszechne jest granie w gry strategiczne, często określane mianem Real Time Strategy. Istnieje spore grono osób, dla których jest to ulubiony sposób spędzania wolnego czasu. Rozwój techniki sprawił, że obecnie powstające gry stały się bardziej atrakcyjne pod względem graficznym, oraz dają znacznie więcej możliwości pod kątem samej rozgrywki. Z całą pewnością ma to pozytywny wpływ na odbiór nowych tytułów oraz popularność gatunku.

Jednym z często pojawiających się problemów przy tego typu grach, gdzie zasady i cała mechanika gry jest z góry określona, jest problem, iż dany gracz może zechcieć zmienić lub określić własny zestaw zasad według których toczy się rozgrywka. Brak takiej możliwości w dłuższej perspektywie może skutkować znudzeniem oraz niechęcią do gry, czego nie raz doświadczyliśmy grając z naszymi znajomymi.

Wychodząc naprzeciw takim potencjalnym problemom postanowiliśmy stworzyć silnik do gier czasu rzeczywistego, która tak jak typowa gra pozwoli na uruchomienie predefiniowanego zestawu zasad, ale jednocześnie pozwoli

graczom na łatwe określenie własnych zasad rozgrywki. Naszą ideę częściowo zaczerpnęliśmy z tego jak działa światowa ekonomia i jakie są relacje pomiędzy poszczególnymi państwami. Każde z tych państw setki i tysiące lat temu zaczynały od pewnej bardzo niewielkiej puli zasobów. Rywalizacja, sojusze oraz wojny sprawiły, że niektóre z nich wyrosły na potęgi, zaś inne przeszły do historii. Co więcej, ta rywalizacja trwa nadal i jest obecna na wielu płaszczyznach - między państwami jak i między ludźmi. Nasz silnik będzie symulował podobny proces rozwoju i rywalizacji - budowania miasta, z tą różnicą, że gracz będzie w stanie sam określić zasady według których ma się toczyć rozgrywka.
Uważamy, że będzie to niezwykła aplikacja, ponieważ będzie ona ciekawym rozwiązaniem i może znaleźć uznanie w oczach graczy. Na dzień dzisiejszy nie znaleźliśmy odpowiednika który dawałby znaczące możliwości modelowania tych zasad. Chcemy aby z naszej projektowanej aplikacji korzystali gracze nie tylko w celach rozrywkowych czy towarzyskich, ale także Ci, którzy dodatkowo liczą na realizm rozgrywki. Dzięki możliwości budowania gier, każdy będzie mógł określić zasady, które mu najbardziej odpowiadają.

Lista potrzeb:

- użytkownik może korzystać z silnika gry ekonomicznej tworzącego grę oraz kontrolującego rozgrywkę,
- użytkownik może uruchomić prostą wizualizację poprzez użycie interfejsu graficznego pozwalającego między innymi na przegląd posiadanych zasobów i domen oraz przedstawiającego możliwe do wykonania akcje,
- gra wykorzystuje element sztucznej inteligencji w roli nadzorcy rozgrywki, odpowiedzialnego za dodatkowe elementy rozgrywki takie jak giełda,
- gra oferuje interaktywny tutorial, który będzie wyświetlał podpowiedzi dla gracza podczas rozgrywki,
- gra oferuje rozgrywkę jednoosobową,
- celem rozgrywki jest odpowiednie użycie zasobów początkowych aby rozbudować startowe miasto, rozwijać posiadany kapitał i zdobywać nowe zasoby,
- realia rozgrywki dotyczą niedalekiej przyszłości, gracz jest odpowiedzialny za utrzymanie przydzielonych mu terenów,

- gracz może korzystać z głównych obiektów rozgrywki - budynków, mieszkańców, zasobów, łańcuchów produkcyjnych i drzewek technologicznych,
- użytkownik ma dostęp do kilku "poziomów" mapy czyli mapy lokalna i mapa globalna (HUD)

Cechy charakterystyczne problemu:

Aplikacja ma za zadanie stworzenie gry na podstawie istniejących plików konfiguracyjnych, które określają zasady według których ma się odbywać rozgrywka, a następnie utrzymanie tej gry do końca rozgrywki. Ponadto aplikacja będzie sterowała zachowaniem sztucznej inteligencji.

Ustalanie reguł - problemem w tym przypadku jest takie dobranie mechanizmu kreatora, aby tworzone reguły wpisywały się w ustalony przez zespół protokół, a jednocześnie dawały pewien zakres elastyczności. Należy te reguły przechowywać oraz ładować do odpowiednich rozgrywek. Interfejs kreatora powinien być intuicyjny, a zarazem poprawny - zależności powinny obejmować tylko zasoby faktycznie utworzone, dodatkowo wprowadzamy hierarchię elementów: przykładowo, mając robotnika, przy określonych w kreatorze regułach możemy go awansować na uczonego, ale nie odwrotnie. Technologię możemy wynaleźć tylko w przypadku posiadania innego zasobu lub wcześniejszego odkrycia innej technologii. Budynek można rozbudować tylko z bazowego budynku. Problemem może okazać się niewłaściwe utworzenie reguły, które nie wpisuje się w dany protokół. Należy do tej sytuacji albo nie dopuścić, albo ją wykryć i poinformować użytkownika odpowiednim komunikatem (np. zasób jest redundantny i nigdzie nie wykorzystywany; budynek budowany za darmo, lub nie posiadający żadnej funkcji ani wpływu na rozgrywkę, bez typu mieszkańców którzy w nim pracują, nie wytwarzający ani nie konsumujący żadnych zasobów; mieszkaniec nie przyporządkowany do żadnego budynku).

Zapewne będzie istniała konieczność wprowadzenia elementów graficznych, które pozwolą w łatwy sposób zarządzać grą oraz sprawią, że całość stanie się bardziej czytelna. Z punktu widzenia gracza, korzystanie w takiej grze z konsoli jest niesamowicie uciążliwe. Aby wyjść naprzeciw tym wymaganiom, nasza aplikacja powinna posiadać interfejs graficzny wizualizujący toczącą się

rozgrywkę. Z tym wiążą się kolejne rzeczy: interfejs powinien dokonać weryfikacji, czy akcje gracza nie przeczą zdefiniowanym zasadom (np. budynek, na który nie ma się dość zasobów nie powinien być dostępny do budowy). Dodatkowo nie wolno stawiać dwóch budynków na kolidujących przestrzeniach. Interfejs, oprócz ekranu rozgrywki, powinien dostarczać zestawu paneli: menu, drzewka technologiczne i zasobów, panel bilansu zasobów oraz panel giełdy i panel samouczka (wskazówek co zrobić). Problemem będzie realizacja reprezentacji mapy - nawigacja po niej powinna być kompromisem pomiędzy prostotą implementacji, a wygodą użytkowania. Przy przesuwaniu widoku, należy zadbać o poprawność i spójność wczytywanego obszaru (wczytujemy budynki, które na pewno istnieją w danym węźle grafu mapy) i jego adekwatność do HUD-a. Przy stawianiu budynków, należy odpowiednio zmodyfikować logikę bilansu - wraz z trwaniem rozgrywki, modyfikatory wynikające z infrastruktury mapy (ilość mieszkańców, stopnie budynków) powinny odzwierciedlać stan portfela gracza i bilans jego zasobów. Samo stawianie budynków jest sytuacją, która rodzi kolejny problem: należy wybrać odpowiedni element interfejsu reprezentujący obiekt, przeciągnąć go na interesujące nas miejsce, dokonać weryfikacji, czy można faktycznie go tam postawić, i ostatecznie zaktualizować inne panele w oparciu o informacje pochodzące z reguł.

Problemem może być sytuacja, w której użytkownik zechce skorzystać z obiecanej symulacji giełdy. Musi się to spotkać z adekwatną i spójną generacją danych, wprowadzającą jednakże elementy losowości, które nie dadzą się łatwo przewidzieć. W ten sposób łatwo urozmaicić rozgrywkę w taki sposób, że użytkownik nieprędko się znudzi. Kursy towarów do wymiany powinny mieć osobne panele i wykresy. Tutaj należy tak zrównoważyć algorytm generowania cen, by z jednej strony nie ułatwiał gry, a z drugiej jej kompletnie nie uniemożliwiał. Gdy ceny będą na okrągło rosły, gracz może się zniechęcić i porzucić ten tryb zabawy. Gdy ceny będą na okrągło spadały, gracz wykupi ogromną ilość zasobów za bezcen. Problem polega na tym, by zbilansować giełdę w ten sposób, aby gracz poczuł się jak prawdziwy inwestor, wykupując towary po najniższej cenie i sprzedając je po możliwie najwyższej, nie czując się przy tym pokrzywdzonym czy oszukanym przez twórców. Prócz tego, symulacja

giełdy musi być oparta o faktycznie zdefiniowane zasoby. Należy zadbać o kooperację modułu giełdy z kreatorem reguł.

Co się stanie w momencie, gdy użytkownik przejdzie do trybu rozgrywki? Istnieje szereg powiązanych z tym zdarzeń. W pierwszej kolejności reguły muszą już być zdefiniowane i dostarczone w pewien ustandaryzowany sposób, gdyż to one wpływają na sposób w jaki budynki, mieszkańcy i zasoby wyglądają, jak się zachowują i jakie zachodzą pomiędzy nimi relacje i hierarchie. Dopiero mając te informacje, algorytm sztucznej inteligencji może decydować, co wolno a czego nie graczowi. Budynki kosztują utrzymanie zdefiniowane w regułach. Ich wyburzenie przynosi ułamek kwoty włożonej w wybudowanie obiektu. To wszystko musi być ostrożnie utrzymywane, zarządzane i wyświetlane w spójnej postaci w panelu bilansu (nie może dojść do sytuacji, gdy odnotujemy wyburzenie tego samego budynku dwukrotnie lub fakt wyburzenia budynku pozostanie zupełnie nie odnotowany).

Częstym problemem może być sytuacja, gdy gracz będzie chciał zrezygnować z dalszej gry. Menu gry powinno oferować możliwość zapisu postępów. Przychodzi pytanie, czy umożliwiać zapis postępów na życzenie klienta jako opcja w menu, czy co określoną ilość czasu - czyli punkty kontrolne. Należy pogodzić oba wymienione podejścia. Aplikacja powinna się zamknąć poprawnie, zadbać o dokładne zwolnienie zasobów.

Dodatkowym problemem może być zdarzenie, gdy użytkownik zechce przerwać rozgrywkę z losowych przyczyn bez wychodzenia całkowicie z aplikacji. Powinno się to odbywać płynnie - stan rozgrywki powinien zostać przywrócony idealnie taki, jakim się go zostawiło.

Propozycje na przyszłe rozbudowanie aplikacji:

Dodatkowym elementem gry może być wprowadzenie systemu walki na zasadzie bezpośrednich bitew lub wojen handlowych. Będzie to wymagało dopracowania silnika gry o kolejne elementy, ale w przyszłości ten dodatek może stać się podstawą rozgrywki wieloosobowej.

Ciekawym dodatkiem, nie wchodzącym w skład trzonu problemu, byłaby aplikacja na platformę mobilną, umożliwiająca sprawdzenie, postępów, osiągnięć i opinii na temat udostępnionych map.

Kolejnym dodatkiem może być możliwość rejestrowania przebiegu gry w postaci zapisywania jej wyników i statystyk. Oczywiście dla różnych reguł gier mogą istnieć różne systemy punktacji. W związku z tym aplikacja powinna mieć możliwość przełączania profili pomiędzy różnymi grami lub możliwość zainstalowania nowego profilu gry poprzez odpowiednie rozszerzenie. Jednak nie będziemy zajmować się w tym w dużym stopniu, a zaimplementujemy jedynie podstawową funkcjonalność którą będzie można rozszerzyć później.

Innym dodatkiem, który może być wprowadzony w późniejszych fazach rozwoju gry i rozszerzony później jest rozgrywka wieloosobowa. Jest to opcja przyszłościowa, wykonaniem której nie będziemy się zajmować. Jest ona przewidziana jedynie po ukończeniu implementacji całości silnika, gdy będzie zaprogramowana cała funkcjonalność , wszystko będzie przetestowane i będzie działało, jak należy. Tak jak zaznaczono powyżej, głównym czynnikiem wpływającym na brak dostarczenia tej funkcjonalności jest niedostateczna ilość czasu.

Podsumowanie:

Głównym problemem przedsięwzięcia wydaje się być interfejs użytkownika: nie wliczając mapy, wizualizacja paneli wymagać będzie sporych nakładów pracy. Szczególną uwagę zespołu należy przykładać do synchronizacji modelu i widoku aplikacji, gdyż widać, że jedno wpływa w dość istotny sposób na drugie, zaś bez spójności danych może dojść do niebezpieczeństwa uszkodzenia logiki modułów.

Wizja rozwiązania

Gra polegająca na tworzeniu strategii ekonomicznej tak, aby uzyskać jak najkorzystniejszy bilans zysków i strat - dzięki temu rozgrywka może być niełatwa do przewidzenia, ale za to fabuła jest ciekawa przy każdym podejściu ze

względu na ciągłą konieczność planowania rozwoju i inwestycji, oceny opłacalności, ryzyka oraz poniesionych szkód. Aplikacja będzie dostępna początkowo dla graczy korzystających z PC.

Menu rozgrywki będzie jądrem całego systemu z dynamicznie ładowanymi komponentami - z jednego kontekstu będzie można z łatwością przełączyć się na inny w granicach zdefiniowanego grafu czynności. Komponenty te to: kreator reguł, tutorial, rozgrywka. Moduły będą oferować szereg funkcjonalności, z których główną będzie kreacja reguł oraz sama rozgrywka.

Gra posiadająca prosty interfejs graficzny - dzięki temu łatwiej jest się zapoznać z jej możliwościami, a jednocześnie gra nie będzie zanadto wymagająca pod względem sprzętowym - nie będzie konieczne posiadanie mocnej karty graficznej i innych zasobów komputerowych. Szata wizualna nie będzie predefiniowana przez twórców: elastyczność systemu pozwala na wprowadzanie dowolnych, dwuwymiarowych tekstur. Niewykluczone, że reprezentacja wizualna obiektów będzie się koncentrowała wyłącznie na dwuwymiarowych teksturach. Uzasadnieniem jest to, że główna przyjemność z rozgrywki polegać będzie na samej logice gry i zasadach nią rządzących, zaś ograniczenie efektów graficznych pozwala na uruchomienie produktu nawet na najsłabszych maszynach.

Istotnym elementem systemu będzie tutorial pokazujący możliwości rozgrywki. Będzie on w postaci tekstowo-graficznej, realizowany w postaci podpowiedzi wyświetlających się na ekranie, lub panelu tekstów, do którego można się dostać z menu gry.

Tworzenie nowej gry będzie polegało na dostarczeniu plików odpowiedniego formatu. Moduł odpowiedzialny za kontrolowanie procesu tworzenia reguł zajmie się ich tworzeniem.
Przewiduje się, jak zaznaczono powyżej, elastyczność w tworzeniu zależności, lecz ogranicza się ona do zdefiniowania trzech typów bytów: mieszkańców, zasobów i budynków, każdy z osobną zakładką i prostymi tekstowymi polami. Dla każdego zbioru powiązań zostanie utworzony nowy kontekst. Możliwe będzie wczytanie kontekstu, pod warunkiem, że będzie poprawnie reprezentowany.

Interfejs będzie prostą tabelą, nadzorowaną przez moduł kreatora. Kolumny tej tabeli będą odczytywane, przetwarzane i łączone w jedną całość gotową dla modułu ładowania. Kolumny tabeli będą definiować odpowiednie atrybuty bytów mieszkańców, budynków i zasobów.

Będą one miały odpowiednie nazwy:
Dla mieszkańców: nazwa, przypisany budynek, poprzednik, następnik, potrzeby(*).
Dla zasobów: nazwa, poprzednik, następnik, budynek
Dla budynków: nazwa, poprzednik, następnik, mieszkaniec-nazwa,mieszkaniec-ilość, zasób, wytwarza(*), pobiera(*), ścieżka do tekstury, ścieżka do ikonki, typ(mieszkalny/przemysłowy).

Dla poprzedników i następników nazwy muszą być istniejące w kontekście, to samo odnośnie budynków i mieszkańców.

Dodatkowo każda z tabel posiadać będzie przycisk "tutorial" prowadzący do panelu obszaru tekstowego, którego treść będzie wpisem w manualu użytkownika. Z zależności zdefiniowanych następników i poprzedników, odpowiednie wskazówki będą generowane podczas rozgrywki w postaci wyskakujących zakładek.

Atrybuty oznaczone (*) będą prowadziły do kolejnych paneli:
Dla mieszkańców: potrzeby prowadzą do widoku dwukolumnowej tabeli, w której kolejne wiersze definiują nazwy zasobów do konsumpcji, oraz szybkości konsumpcji przez jednego mieszkańca w jednostkach na sekundę.
Dla budynków: “wytwarza” i “pobiera”: dwukolumnowe tabele nazwa zasobu i szybkość odpowiednio operacji wytwarzania bądź pobierania w jednostkach na sekundę. Niewykluczone zdefiniowanie obu naraz. Moduł kreatora dopilnuje, żeby budynek nie wytwarzał tego samego co pobiera.

Po odpowiednim wypełnieniu pól w kreatorze, użytkownik klika "stwórz". Jeśli operacja powiedzie się, następuje powrót do menu i odpowiednie pliki są tworzone. W przeciwnym przypadku, komunikaty dotyczące błędów wyświetlają

się na obszarze tekstowym panelu do momentu, w którym wszystkie błędy zostały naprawione przez twórcę.

Użytkownik uruchamiając program będzie miał możliwość uruchomienia predefiniowanej gry, a także stworzenia nowych gier z innymi zależnościami. Po wybraniu gry silnik na podstawie plików konfiguracyjnych tworzy i uruchamia grę. Wszystkie zależności zdefiniowane są wstrzykiwane w rozgrywkę za pomocą modułu wczytywania, jednocześnie tworzone są odpowiednie struktury w modelu. Ikonki zasobów, budynków, mieszkańców, znalezione na podanych przez użytkownika ścieżkach, pojawiają się w odpowiednich miejscach w panelach interfejsu. Jednocześnie z tym modułem ruszają moduły giełdy i tutoriala, załadowane do modułu menu. Po wciśnięciu pauzy, pojawiają się dodatkowe opcje, w tym "opuść rozgrywkę", "zapisz rozgrywkę" itp.

Rozgrywka będzie przebiegała w następujący sposób: Gracz zaczyna od małej osady z ograniczonymi zasobami, którą stopniowo będzie rozwijał. Startowym budynkiem jest zawsze Siedziba. Gra toczy się w niedalekiej przyszłości - świat gry jest częściowo znany użytkownikowi z życia codziennego, ale zarazem pobudza wyobraźnię. Użytkownik będzie zmuszony do zarządzania swoimi zasobami w taki sposób, aby zmaksymalizować funkcję celu - czyli szybkość rozwoju swojego miasta. Powoli rozwija swoją domenę przez tworzenie nowych budowli oraz gromadzenie coraz większej ilości zasobów. Wraz z rozwojem odblokowywane będą nowe możliwości rozwoju zgodnie z drzewkami technologicznymi oraz łańcuchami produkcyjnymi, które określają jakie elementy są wymagane, aby odblokowane zostały elementy bardziej zaawansowane. Gracz może na bieżąco zmieniać swoją strategię, dostosowując ją do zmieniających się warunków. Taką decyzją może być dokonywanie wymian zasobów na giełdzie, budowa odpowiednich budynków czy rozwój odpowiedniej technologii. Głównym celem jest jak najefektywniejszy rozwój swojej domeny. Gra toczy się dopóki gracz nie opuści rozgrywki.
Jeśli dany gracz będzie chciał opuścić grę naciska przycisk "Opuść grę". Jeśli gracz zdecyduje, że chce sobie zrobić przerwę, to wciska przycisk "Pauza".

Ekran rozgrywki będzie warstwowy: główne okno otoczone zostanie kilkoma panelami, z których można będzie przemieszczać się pomiędzy

widokiem mapy, widokiem tutoriala, widokiem giełdy, widokiem HUD-a, czy widokiem bilansu zasobów. Wszystkie będą korzystać ze scentralizowanych struktur danych, których wartości będą odpowiednio wpływać na to, co jest wyświetlane na ekranie.

Główny ekran mapy będzie trzonem systemu. To stąd gracz będzie podejmować decyzje wpływające na resztę paneli. Czynnością najczęściej wykonywaną będzie budowanie, burzenie bądź ulepszanie budynków. W momencie, gdy gracz będzie chciał postawić jakiś budynek, będzie musiał najechać myszą na reprezentującą go ikonę - po wybraniu ikony, zostaną zweryfikowane warunki, które muszą być spełnione, aby budynek mógł być poprawnie wyselekcjonowany. Pomyślny test zostanie zwizualizowany "widmowym" budynkiem przyczepionym do kursora, który pozostanie tam, aż do ponownego kliknięcia na obszarze mapy. Wybrany fragment mapy zostanie zbadany pod kątem wolnej przestrzeni, dopiero wtedy stan modelu jest aktualizowany i co sekundę nowo wybudowany obiekt wpływa na rozgrywkę zgodnie ze zdefiniowanymi w kreatorze zasadami. Mini reprezentacja obiektu pojawia się na radarze HUD.

Aby wyburzyć obiekt, należy z ekranu mapy wybrać ikonę "usuń". Gracz najeżdża wtedy kursorem na fragment mapy, który jest badany pod kątem istnienia tam jakiegokolwiek obiektu. Po kliknięciu, jeśli na kawałku obszaru istnieje budynek, podejmowany jest ciąg akcji: reprezentacja obiektu znika z mapy i HUD-a, jego wpływ na system zostanie usunięty, a gracz otrzyma część zainwestowanych w budynek zasobów.

Gracz może także poruszać się po mapie, cztery kierunki świata są reprezentowane poprzez ikonki przy odpowiednich krawędziach ekranu. Po kliknięciu, gracz przenosi się do kolejnego fragmentu mapy, o czym zostanie poinformowany HUD, zmieniając swój widok odpowiednio.

Gracz może wyświetlić informacje o budynku. W tym celu najeżdża nań kursorem i klika LPM. Informacje wyświetlane są w przyjaznej formie w postaci notatki.

Użytkownik może w każdej chwili rozpocząć wykupywać lub sprzedawać towary na giełdzie. Po naciśnięciu odpowiedniego przycisku w panelu interfejsu, zostaje przeniesiony do trybu giełdy - zastępuje on widok mapy, a rozgrywka

jest zatrzymana. Pojawia się wybór towarów do kupienia, zgodnie z dostarczonymi regułami kreatora. Po kliknięciu na ikonę towaru, gracz widzi wykres jego kursu akcji. Może podjąć decyzję o kupnie lub sprzedaży, albo powrócić do menu mapy. Jeśli zostanie dokonana jakakolwiek transakcja, modyfikowany jest wirtualny magazyn i zasoby gracza.

Moduł tutoriala jest odpowiedzialny za przyjazne wprowadzenie gracza w mechanikę rozgrywki. Tekst wyeksportowany z panelu kreatora jest tutaj przechowywany w postaci poukładanych kart manuala. Użytkownik, po kliknięciu na interesujący go wpis, widzi wyeksportowaną treść w obszarze tekstowym, ewentualnie z odpowiednią ilustracją pochodzącą z kreatora. Moduł ten odpowiedzialny jest także za wyświetlanie wskazówek na ekranie mapy.

Dane gry są zapisywane lokalnie - daje to możliwość kontynuowania zapisanej sesji gry w innym czasie.

Korzyści dla klienta:

- niezapomniane przeżycia oraz dobra zabawa, które można współdzielić wraz z przyjaciółmi,
- stymulacja zmysłu strategicznego oraz ekonomicznego nabywcy,
- nostalgiczny "powrót do przeszłości" - dorobek gatunku ukazany w nowym świetle,
- gra, w którą będzie chętnie grało szerokie grono użytkowników,
- szeroka gama możliwości: zależności pomiędzy zasobami, budynkami i mieszkańcami od najbardziej hardcorowych po niezbyt skomplikowane - według zasady "sky is the limit",
- interfejs przyjazny użytkownikowi, ale zarazem nie wymagający sprzętu najnowszej generacji,
- aplikacja z eleganckim kodem i dokumentacją, aby w przyszłości kolejni programiści mogli ją z powodzeniem rozwijać,
- silnik umożliwiający stworzenie gry która wymaga logicznego myślenia i umiejętności tworzenia strategii, która da potencjalnie najlepsze wyniki nawet w nie całkiem sprzyjających sytuacjach

Typowa działalność użytkownika - podsumowanie:

- tworzenie własnych reguł i zależności,

- rozgrywka na przygotowanych w przykładowym tutorialu zasadach

Typowe procedury użytkownika - podsumowanie:

- ładowanie modułów,
- wydawanie systemowi poleceń,
- utrzymywanie domeny

Usługi realizowane przez system - podsumowanie:

- logika gry,
- wizualizacja rozgrywki,
- tutorial,
- utrzymanie informacji pochodzących od graczy,
- konstruktor reguł

Udogodnienia, które gotowy produkt przedstawi użytkownikom:

- kiedy użytkownik będzie chciał rozeznać się w systemie, system załaduje tutorial,
- w momencie, gdy użytkownik zechce przeglądnąć swoje osiągnięcia, zostaną one wyświetlone w przyjaznej formie przez system,
- w momencie, gdy użytkownik zechce stworzyć własny scenariusz, system załaduje odpowiednią funkcjonalność,
- kiedy użytkownik będzie chciał zaplanować swoją strategię na dalszą część rozgrywki, przedstawione zostaną drzewka technologiczne oraz łańcuchy produkcyjne

Zestaw wymagań niefunkcjonalnych:

Technologie:

- aplikacja przeznaczona na systemy operacyjne Windows
- Java 8, ponieważ nasi fachowcy czują się najpewniej w tym języku programowania, projekt od strony implementacyjnej będzie wyglądał czytelniej, a co za tym idzie - zmiany wprowadzane do systemu będą łatwiej zarządzalne, a system jako całość będzie prostszy w utrzymaniu, rozwoju i konserwacji,

- Python 3, w szczególności biblioteka Pygame - wraz z wbudowanym OpenGL, narzędzie to idealnie nada się do tworzenia prostego interfejsu graficznego użytkownika,
- wykorzystanie języków programowania takich jak Java i Python zapewnia większą elastyczność platformową oraz łatwiejszy rozwój projektu w przyszłości - oba języki należą do bardzo popularnych i dość dobrze udokumentowanych,
- dane będą przechowywane w formacie .json ze względu prostotę i jego popularność

Koncepcja systemu

Wstęp

Tworzona aplikacja będzie działała zgodnie z koncepcją silnika gry. Sam silnik jest programem lub biblioteką implementującą zasadniczą funkcjonalność (logikę) aplikacji. W przypadku silnika gry mamy do czynienia z główną częścią kodu gry komputerowej, która zajmuje się interakcją elementów gry. Może mieć w sobie wbudowane moduły odpowiedzialne za poszczególne aspekty rozgrywki. Nasz silnik będzie umożliwiał tworzenie zasad gry oraz prowadzenie samej rozgrywki według określonych parametrów.
Formalnie użytkownikami naszej aplikacji będą twórcy gier którzy zajmują się kreacją zasad gry oraz sami gracze którzy biorą udział w rozgrywce przy użyciu wcześniej określonych zasad. Gracz otrzymuje od twórcy pliki konfiguracyjne które wykorzystuje do stworzenia gry i prowadzenia rozgrywki. Bardzo ważny jest fakt, że twórca i gracz może być jedną i tą samą osobą. Zatem użytkownik może stworzyć zasady gry, a następnie na ich podstawie grę w której będzie brał udział.
Użytkownik nie będzie mógł wykreować dowolnej gry za pomocą naszego silnika, jest to silnik wyspecjalizowany do tworzenia gier typu Real Time Strategy. Precyzyjniej, użytkownik będzie miał możliwość kreacji gier według zasad określonych przez moduł kreatora. Przykładowo, będzie możliwość zmiany budynków możliwych do postawienia, ale nie będzie możliwe zdefiniowanie jednostek wojskowych, ponieważ silnik ich nie uwzględnia w swoim działaniu. Oczywiście, jeżeli użytkownik bardzo będzie tego chciał, może dopisać takie funkcjonalności - nasza aplikacja będzie otwarta na modyfikacje i dodatki.

Architektura

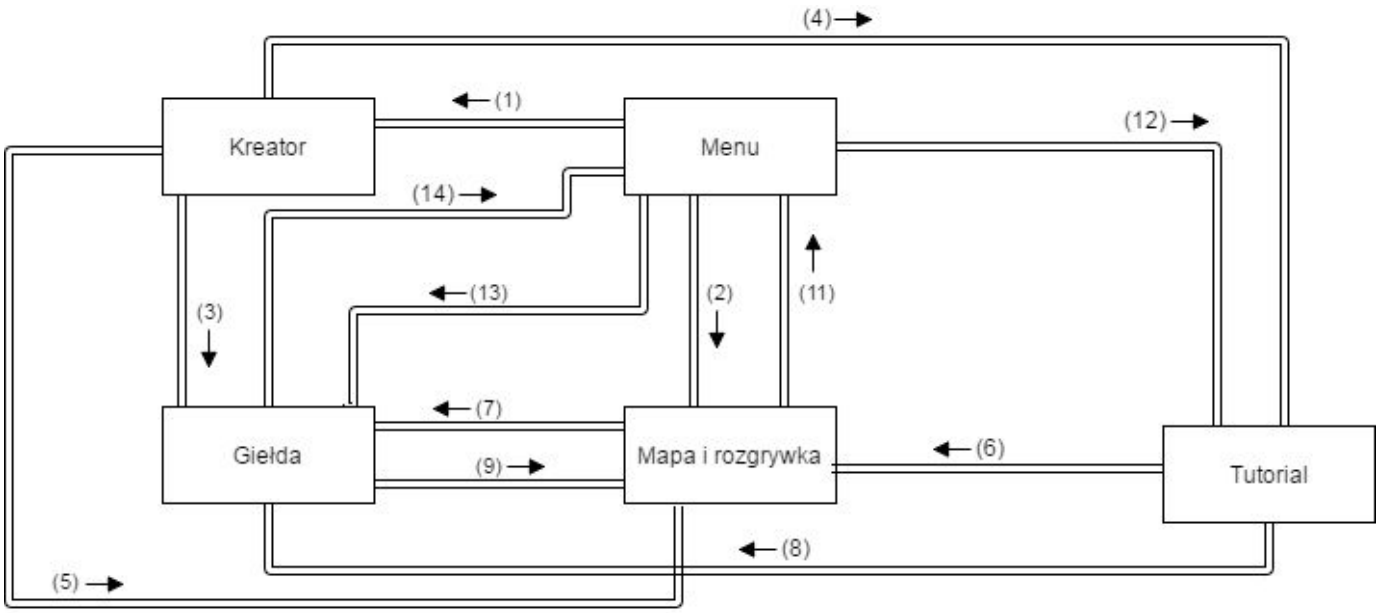
System składa się z następujących modułów:

- moduł menu,
- moduł mapy i rozgrywki (w tym HUD),
- moduł kreatora - zapis i odczyt reguł oraz zależności,
- moduł giełdy,
- moduł tutoriala

Spis funkcji systemowych:

1. CreateDependencies - zapisuje zależności do pliku
2. QuitGame - kończy rozgrywkę
3. LoadDependencies - ładuje zależności Kreatora z pliku i wstrzykuje do rozgrywki
4. StartGame - daje wybór zasad do wczytania, tworzy rozgrywkę z załadowanych przez LoadDependencies zależności
5. ShowHints - ładuje wskazówki tutoriala
6. GenerateTradeMarket - stwórz wirtualne kursy zasobów
7. ChangeView - zmienia widok (np. z mapy głównej do menu)
8. UpdateModel - zmienia model po wykryciu aktywności gracza
9. CreateBuilding - tworzy budynek i wszystkie związane z nim zależności; inicjuje wywołanie UpdateModel
10. DeleteBuilding - usuwa budynek i wszystkie związane z nim zależności; inicjuje wywołanie UpdateModel
11. SaveGame - zapisuje rozgrywkę
12. LoadGame - wczytuje rozgrywkę
13. Init - inicjuje kernela systemu, alokuje struktury danych, tworzy komunikację pomiędzy modułami.

Architektura logiczna



***Rys 1**. Diagram komunikacji*

Użytkownik włącza aplikację i zostaje wyświetlony interfejs modułu Menu. Następnie użytkownik wybiera opcję rozpoczęcia rozgrywki. Moduł Menu prosi wtedy o podanie ścieżki do zestawu reguł jaki ma zostać załadowany. Po sprawdzeniu poprawności wprowadzonych danych, moduł Menu tworzy zlecenie załadowania zestawu reguł, w którym zawiera ścieżkę do nich, i wysyła je do modułu Kreatora (1).

Moduł Kreatora wczytuje i sprawdza poprawność reguł gry. Po wczytaniu danych przekazuje je do odpowiednich modułów, które na ich podstawie będą wyświetlały odpowiednie tekstury oraz zarządzały rozgrywką: do Giełdy przekazuje informacje dotyczące zasobów (3), do Tutoriala przekazuje tekst, który zostanie wykorzystany do tworzenia podpowiedzi dla gracza (4), do modułu Map i Rozgrywki przekazuje ogólne zasady dotyczące rozgrywki (5), m.in. informacje o budynkach, zasobach, mieszkańcach, mapie i występujących w świecie ograniczeniach.

Kiedy zasady gry są już załadowane, moduł Menu tworzy zlecenie wyświetlenia początkowej osady i przekazuje je do modułu Map i Rozgrywki (2).

Podczas rozgrywki wyświetlane są podpowiedzi dla gracza. Za ich generowanie odpowiedzialny jest moduł Tutorial, który tworzy krótką wiadomość tekstową i przekazuje ją do modułu Map i Rozgrywki (6), który następnie wyświetla ją na ekranie.

Podpowiedzi są również wyświetlane, kiedy gracz znajduje się w panelu giełdy. Moduł Tutorial generuje krótką wiadomość tekstową, którą następnie przekazuje do modułu Giełdy (8), który następnie wyświetla odebrane dane.

Gracz może przejść z panelu rozgrywki do panelu giełdy. Odbywa się to poprzez wygenerowanie przez moduł Map i Rozgrywki zlecenia wyświetlenia panelu giełdy i przesłaniu go do modułu Giełdy (7).

Oczywiście gracz może powrócić z panelu giełdy do panelu głównej rozgrywki. Odbywa się to poprzez wygenerowanie przez moduł Giełdy odpowiedniego zlecenia i wysłanie go do modułu Map i Rozgrywki (9).

Kiedy użytkownik chce przejść do menu głównego moduł Map i Rozgrywki generuje zlecenie wyświetlenia odpowiedniego panelu i przekazuje je do modułu Menu (11).

Z menu rozgrywki użytkownik może dostać się do tutoriala w formie panelu tekstów. Moduł Menu generuje wtedy zlecenie wyświetlenia tutoriala i wysyła je do modułu Tutorial (12).

Z menu rozgrywki użytkownik może również przejść do zapisania postępów gry. Użytkownik wybiera odpowiednią opcję z menu, a następnie proszony jest o podanie nazwy zapisu. Po sprawdzeniu poprawności wprowadzonych danych moduł Menu generuje komunikaty do modułów Giełdy (13) oraz Map i Rozgrywki (2), które zawierają zlecenie przekazania danych potrzebnych do stworzenia pliku z postępami w grze. W odpowiedzi moduł Giełdy (14) oraz moduł Map i Rozgrywki (11) generują komunikaty z danymi potrzebnymi do utworzenia zapisu gry.

Użytkownik może również zechcieć załadować zapisaną wcześniej grę. W tym celu, uruchamia aplikację i wybiera z menu odpowiednią opcję. Następnie moduł Menu prosi użytkownika o wybranie pliku z zapisem. Moduł Menu ładuje wybrane dane, oraz generuje komunikaty dla modułu Giełdy (13) oraz dla modułu Map i Rozgrywki (2), w których zawiera dane potrzebne tym modułom do odtworzenia stanu rozgrywki. Następnie moduł Menu generuje zlecenie wyświetlenia panelu rozgrywki i przekazuje je do modułu Map i Rozgrywki (2).

Z menu gry użytkownik może również przejść do tworzenia zasad gry. W tym celu moduł Menu generuje komunikat, zawierający zlecenie przejścia do edytora reguł. Komunikat ten jest następnie wysyłany do modułu Kreatora (1), który uruchamia edytor.

Plan rzeczowy

Zadanie nr 1: Sporządzenie podstawowego szkieletu systemu

Opis: Zadanie będzie polegać na nadaniu odpowiedniej struktury naszemu projektowi, tak, aby zminimalizować wysiłek przełączania się pomiędzy różnymi trybami/widokami produktu. W skład szkieletu będzie wchodziła dynamiczna struktura spinająca moduły, ich funkcjonalności i powiązania na zasadzie magistrali (sposób komunikacji i wymieniania globalnych informacji), dostarczająca logiki kontroli (przełączania pomiędzy spiętymi logikami-widokami trybów na życzenie gracza/twórcy).
Przewidywany czas wykonania: 40 roboczogodzin

Zadanie nr 2: Utworzenie Kreatora Zależności

Opis: Zadanie będzie polegać na sporządzeniu logiki i widoku modułu, dostarczającego symetrycznych względem siebie interfejsów zapisywania i wczytywania zależności. Element tworzący będzie odpowiedzialny za wczytywanie opisu zasobów i na tej podstawie utworzenia ich logicznej reprezentacji. Działania w tym zakresie będą uwzględniały implementację widoku, a także strażnika pilnującego, by pewne granice tworzenia zostały

zachowane. Dodatkowo zaimplementowana zostanie funkcja zapisująca dane w odpowiednim formacie. Tworzone będą tutaj klasy fabryk obiektów.
Interfejs wczytujący będzie zaimplementowany w sposób udostępniający utworzone struktury dla reszty systemu. Udostępniony zostanie globalny Reprezentant Obiektów, informujący o utworzonych zależnościach.
Przewidywany czas wykonania: 20 roboczogodzin

Zadanie nr 3: Stworzenie modułu Mapy

Opis: W tym zadaniu należy skupić się na implementacji modułu będącego głównym elementem wpływającym na resztę systemu. Zaimplementujemy mechanizmy pozwalające graczowi wchodzić w interakcję z tym modułem, które będą się różniły od mechanizmów interakcji planowanych dla innych części systemu. Stworzymy zestaw przycisków, które posłużą jako główny sposób komunikacji użytkownika z rozgrywką utworzoną przez silnik.
Zaimplementowane zostaną klasy, których obiekty będą tworzone dynamicznie przez Fabryki Kreatora, reprezentujące konkretne instancje budynków, mieszkańców itp. Dodatkowo w skład wchodzić będzie byt rejestrujący nieustanny wpływ tworzonych przez Fabryki obiektów, aktualizujące globalne struktury. Demon ten będzie nieustannie aktualizował panel widoku(głównie panel bilansu).
Przewidywany czas wykonania: 25 roboczogodzin

Zadanie nr 4: stworzenie modułu giełdy

Opis: Celem będzie implementacja interfejsu rodziny wymienialnych algorytmów generujących dane giełdowe.

Przewidywany czas wykonania: 13 roboczogodzin

Plan pracy

Sądny dzień 1 (12.04.2017)

- sprawdzenie spójności modelu architektury,
- stworzenie modelu logicznego aplikacji,
- dopracowanie pozostałych dokumentów projektu

Cele do osiągnięcia:

- model logiczny umożliwiający pracę w dalszym etapie przedsięwzięcia projektowego,
- szkielety dla poszczególnych klas

Produkty:

- szkielet struktury kodu,
- wstępny moduł menu powiązany logicznie z innymi modułami, ale jeszcze bez funkcjonalności tych modułów

Sądny dzień 2 (17.05.2017)

- weryfikacja stworzonego szkieletu aplikacji pod kątem zgodności z architekturą systemu,
- możliwa zmiana struktury podsystemów w przypadku zauważenia dużych problemów ze spełnieniem obecnych założeń lub zmiany wymagań,
- sprawdzenie i ewentualne poprawki modelu logicznego aplikacji,
- implementacja podstawowych funkcjonalności dla wszystkich modułów,
- testowanie,
- integracja

Cele do osiągnięcia

- użytkownik może uruchomić aplikację,
- użytkownik może stworzyć zestaw podstawowych reguł,
- użytkownik może uruchomić grę z danym zestawem reguł,
- użytkownik może skorzystać z podstawowych funkcjonalności giełdy,
- użytkownik może zatrzymać lub wyjść z gry

Produkty:

- ewentualna poprawa menu,
- moduł kreatora, umożliwiający stworzenie podstawowych zasad,
- moduł mapy i rozgrywki, umożliwiający uruchomienie gry z zestawem zasad,
- wstępne wersje modułu giełdy z podstawowymi funkcjami,
- dokumentacja deweloperska stworzonego kodu.

Sądny dzień 3 (7.06.2017)

- weryfikacja stworzonego szkieletu aplikacji pod kątem zgodności z modelem logicznym i strukturą podsystemów,
- implementacja pozostałych funkcjonalności systemu,
- dopracowanie poszczególnych modułów,
- testowanie,
- integracja

Cele do osiągnięcia:

- Użytkownik może korzystać z pełnej funkcjonalności giełdy z elementami sztucznej inteligencji,
- użytkownik może skorzystać z samouczka w trakcie rozgrywki,
- użytkownik może skorzystać z pełnej funkcjonalności kreatora,
- użytkownik może wczytywać i zapisywać grę

Produkty:

- ostateczna wersja aplikacji,
- ostateczna dokumentacja deweloperska całego kodu,
- komplet dokumentów dotyczących aplikacji - podręcznik instalacji, konfiguracji, przewodnik użytkownika,
- sprawozdanie z przebiegu projektu,
- prezentacja aplikacji

Analiza ryzyka

W poniższej analizie wzięliśmy pod uwagę najbardziej prawdopodobne trudności na które możemy się natknąć podczas tworzenia aplikacji i ich ewentualne skutki na przebieg całego projektu.

Brak znajomości technologii - najprawdopodobniej podczas implementacji aplikacji zetkniemy się z technologią, z która wcześniej nie mieliśmy do czynienia.
Skutkiem może być konieczność poświęcenia większej ilości czasu na projekt.

Dobór złej technologii - istnieje zagrożenie dobrania złych narzędzi do rozwiązania danego problemu technicznego. Po odpowiednim przygotowaniu przed fazą implementacyjną raczej nie powinno do tego dojść.
Skutkiem może być konieczność zmiany technologii lub modyfikacji założeń projektowych.

Zmiana wizji projektu i wstępnych wymagań - ze względu na złożoność projektu i nieprzewidziane problemy projektowe, potencjalnie możliwa jest zmiana pierwotnych założeń dotyczących funkcjonalności aplikacji.
Skutkiem może być otrzymanie końcowego produktu, różniącego się od pierwotnej wizji.

Niewykryte błędy techniczne - przewidujemy testowanie finalnej aplikacji, ale żadne testy nigdy nie dadzą absolutnej gwarancji pełnej poprawności działania aplikacji.
Skutkiem może być aplikacja, która pod wpływem pewnych czynników może się zachowywać w nieprzewidziany sposób.

Problemy z integracją modułów - podczas implementacji może się okazać, że poszczególne moduły będzie trudno ze sobą powiązać ze względów logicznych lub technicznych.
Skutkiem może być opóźnienie w tworzeniu aplikacji lub konieczność modyfikacji pierwotnego modelu aplikacji.

Błędna implementacja niektórych funkcjonalności - istnieje ryzyko, że stworzymy funkcjonalność, która będzie zawierała błędy techniczne, takie jak niepoprawna implementacja algorytmu i wycieki pamięci, lub logiczne, takie jak działanie nie do końca zgodne z oczekiwaniami.
Skutkiem może być konieczność poprawienia lub stworzenia od nowa poszczególnych funkcjonalności.

Słaba optymalizacja działania - możliwe jest, że podczas implementacji skorzystamy z tych mniej optymalnych sposobów rozwiązania danego problemu, głównie ze względu na brak wiedzy o tych innych, alternatywnych sposobach.

Skutkiem może być mniej wydajna aplikacja, niż gdyby skorzystać z lepszych rozwiązań.

Brak realizacji niektórych wymagań funkcjonalnych - przez ograniczony czas realizacji projektu możemy nie być w stanie zrealizować wszystkie wymagania.
Skutkiem może być stworzenie aplikacji, która nie oferuje wszystkich przewidzianych wymagań.

Błędy w dokumentacji - błędy te mogą wynikać z częstych zmian sposobu implementacji danej funkcjonalności. Innym problemem może być niepoprawna struktura lub sposób jej prowadzenia.
Skutkiem może być stworzenie niezrozumiałej dokumentacji lub konieczność napisania całości lub jej fragmentów od nowa.

Dziennik implementacyjny

Sobota 1 kwietnia 2017 roku:

godzina 5:32 - Założono projekt javowy i pythonowy w środowiskach odpowiednio: Eclipse i Sublime.

godzina 5:47 - Utworzono paczki CreatorModule, ExchangeModule, MapModule, MenuModule, TutorialModule.

godzina 6:02 - utworzono wstępną implementację wczytywania z pliku danych komend przekazywanych warstwie widoku przez warstwę kontrolera.

godzina 6:17 - wpisano kilka importów i kawałka kodu pythonowego obsługującego komunikację pomiędzy logiką a widokiem.

godzina 6:47 - dokonano dokończenia implementacji komunikatora po stronie Pythona (w tym parsowanie plików JSON do słowników)

godzina 7:02 - napisano podklasę Sender w klasie Mediator.java

godzina 7:17 - zaimplementowano wstępnie podklasę Receiver w klasie Mediator.java

godzina 7:47 - dokończono debugowanie komunikacji Mediatorów Pythonowego i Javowego

godzina 8:00 - pomyślnie scalono oba Mediatory

godzina 8:20 - Utworzono klasę Node.java

godzina 8:35 - uruchomiono pomyślnie metodę buildDataFromJSON() w klasie Node.java

godzina 9:00 - 10:00 testowano i podejmowano próby analizy pliku PygamePanel.py - skryptu pozwalającego połączyć ze sobą technologie Pygame i wx.

godzina 10:15 - 11:00 testowano i zanalizowano skrypt ConcurrentPanel.py - zrozumiano jak można wykorzystać bibliotekę Pygame w kilku panelach jednocześnie

godzina 11:15 - włączono możliwość łatwego przełączania pomiędzy widokami menu.

godzina 11:33 - utworzono główny framework Widoku: skrypty MapView.py, MenuView.py, ExchangeView.py, CreatorView.py

godzina 11:50 - utworzono panel mapy

godzina 12:05 - wypełniono panel mapy przykładową zawartością

godzina 12:30 - przetestowano przełączanie (na poziomie warstwy widoku) pomiędzy widokami menu i mapy.

godzina 12:45 - utworzono pusty panel widoku giełdy

godzina 13:00 - wypełnianie widoku giełdy przykładową zawartością, napotkano na problem zintegrowania biblioteki Pygame z biblioteką Matplotlib.

godzina 13:30 - rozwiązano wyżej wspomniany problem

godzina 13:45 - przetestowano przykładową zawartość giełdy

godzina 14:00 - wykryto ogromny problem - okazało się, że w momencie gdy jest równocześnie uruchomionych kilka paneli pygame, tylko jeden z nich może być aktywny w danym momencie. Nastąpił kryzys implementacyjny i próby jego rozwiązania.

godzina 14:45 - rozwiązano powyższy kryzys w elegancki sposób, przystąpiono do testowania utworzonych do tego momentu widoków

godzina 15:00 przetestowano przełączanie pomiędzy mapą, menu a Giełdą.

godzina 15:17 - napisano grid dotyczący budynków w widoku Kreatora

godzina 15:32 - napisano grid dotyczący zasobów w widoku Kreatora

godzina 15:47 - napisano grid dotyczący mieszkańców w widoku Kreatora

godzina 15:50 - 16.07 - podejmowano próby wyświetlenia widoku gridu w widoku Kreatora w zależności od tekstu wybranego w okienku u góry panelu -

długość tej fazy implementacji wynikała z wertowania artykułów, tutoriali oraz dokumentacji biblioteki wx.

godzina 16:25 - przetestowano widok kreatora

godzina 16:40 - przetestowano przełączanie pomiędzy widokami Giełdy, mapy, menu oraz Kreatora

godzina 17:03 - wprowadzono element dźwiękowy do widoków; napotkano problem polegający na tym, że pierwszy wyświetlany widok nie odtwarzał muzyki.

godzina 18:04 - wyeliminowano powyższy problem; poradzono sobie z drobnym problemem warstwy audio.

godzina 18:10 - 19:00 gruntowne testowanie zaimplementowanych do tej pory widoków.

godzina 19:02 - koniec prac

Piątek 7 kwietnia 2017 roku:

godzina 14:54 - poprawienie głównego widoku menu

godzina 15:32 - poprawienie widoku kreatora

godzina 16:11 - przetestowanie spójności działania poprawionych widoków

godzina 16:58 - drobne poprawki w elemencie dźwiękowym

godzina 17:39 - koniec prac

Poniedziałek 10 kwietnia 2017 roku:

godzina 8:32 - wprowadzenie poprawek implementacyjnych oraz opisu funkcjonalności poszczególnych klas

godzina 10:43 - koniec prac

Przypadki użycia:

UC-01: Uruchomienie aplikacji
Cel: Użytkownik chce uruchomić aplikację
Kontekst: Użytkownik chce skorzystać z kreatora w celu stworzenia nowego kontekstu lub chce utworzyć nową rozgrywkę
Warunek początkowy: Aplikacja jest wyłączona
Scenariusz główny:
1.Użytkownik wybiera ikonę aplikacji z pulpitu klikając dwukrotnie LPM.
2.Aplikacja wyświetla prosty interfejs graficzny zawierający menu.

Scenariusz alternatywny:

2.1 W przypadku błędu uruchomienia użytkownik otrzymuje stosowny komunikat.

UC-02: Uruchomienie kreatora bytów

Cel: Aplikacja wyświetla kreator bytów

Kontekst: Użytkownik chce zmodyfikować istniejący kontekst lub stworzyć nowy

Warunek początkowy: Użytkownik znajduje się w widoku menu

Scenariusz główny:

1.Użytkownik wybiera z menu opcję „Kreator"

2.Użytkownik wybiera, że chce skorzystać z istniejącego kontekstu.

3.Użytkownik wybiera poprawny plik kontekstu istniejący w systemie plików

4.Aplikacja ładuje dane z pliku

5.Aplikacja wyświetla edytor kontekstu

Scenariusz alternatywny:

2.1.Użytkownik wybiera, że chce utworzyć nowy kontekst. Wówczas następuje przejście do punktu 5.

4.1.W przypadku niepoprawnego formatu pliku użytkownik informuje informację o błędzie i propozycję utworzenia nowego kontekstu.

UC-03: Uruchomienie tutoriala

Cel: Użytkownik chce zobaczyć wskazówki dot. tworzenia bytów

Kontekst: Użytkownik chce się dowiedzieć, w jaki sposób poprawnie zdefiniować byty i zależności między nimi.

Warunek początkowy: Użytkownik znajduje się w jednym z widoków kreatora (typów mieszkańców, zasobów lub budynków)

Scenariusz główny:

1.Użytkownik wybiera ikonę tutoriala

2.Aplikacja wyświetla panel obszaru tekstowego z odpowiednim wpisem w manualu
użytkownika.

Scenariusz alternatywny:

2.1. W przypadku nie znalezienia danego elementu w manualu, użytkownik otrzymuje informację o błędzie.

UC-04: Zdefiniowanie mieszkańców

Cel: Użytkownik chce zdefiniować mieszkańców dla danego kontekstu

Kontekst: Użytkownik chce określić wartości wszystkich atrybutów dla każdego z typów mieszkańców

Warunek początkowy: Użytkownik znajduje się w głównym widoku kreatora

Scenariusz główny:

1.Użytkownik wybiera w kreatorze zakładkę „Mieszkańcy"

2.Użytkownik uzupełnia lub/i modyfikuje istniejące wartości atrybutów

UC-05: Zdefiniowanie zasobów

Cel: Użytkownik chce zdefiniować zasoby

Kontekst: Użytkownik chce określić powiązania pomiędzy zasobami i wszystkie ich atrybuty

Warunek początkowy: Użytkownik znajduje się widoku kreatora

Scenariusz główny:

1.Użytkownik wybiera w kreatorze zakładkę „Zasoby"

2.Użytkownik uzupełnia lub/i modyfikuje istniejące wartości atrybutów

UC-06: Zdefiniowanie budynków

Cel: Użytkownik chce zdefiniować budynki

Kontekst: Użytkownik chce określić wszystkie atrybuty budynków - dla każdego z nich: widok, funkcję, wpływ na świat, typy zasobów z których budynek korzysta,

Warunek początkowy: Użytkownik znajduje się widoku kreatora

Scenariusz główny:

1.Użytkownik wybiera w kreatorze zakładkę "Budynki"

2.Użytkownik uzupełnia lub/i modyfikuje istniejące wartości atrybutów

UC-07: Zapisanie kontekstu

Cel: Użytkownik chce zapisać kontekst

Kontekst: Użytkownik chce zachować zmiany dokonane w kontekście w celu wykorzystania go później w rozgrywce

Warunek początkowy: Użytkownik znajduje się widoku kreatora

Scenariusz główny:

1.Użytkownik zatwierdza zmiany klikając przycisk „Stwórz"

2.Użytkownik wraca do widoku menu

Scenariusz alternatywny:

2.1. W przypadku nieprawidłowych wartości komunikaty dotyczące błędów wyświetlają się na obszarze tekstowym panelu do momentu, w którym wszystkie błędy zostały naprawione przez twórcę.

UC-08: Wybieranie tekstur

Cel: Użytkownik chce wybrać tekstury

Kontekst: Użytkownik wybiera tekstury w celu skonfigurowania stylistyki bytów

Warunek początkowy: Użytkownik znajduje się w widoku kreatora

Scenariusz główny:

1.Użytkownik wybiera opcję "Tekstury"

2.Użytkownik wybiera, dla jakiego rodzaju bytu chce określić teksturę

3.Użytkownik wybiera plik tekstury w obsługiwanym przez aplikację formacie i zatwierdza swój wybór

4.Następuje powrót do kreatora

Scenariusz alternatywny:

4.1. W przypadku problemów z załadowaniem tekstury użytkownik otrzyma stosowny komunikat o błędzie i propozycję wybrania innej tekstury.

UC-09: Stworzenie rozgrywki

Cel: Użytkownik chce stworzyć rozgrywkę

Kontekst: Użytkownik chce wykorzystać istniejący wcześniej kontekst w nowej rozgrywce

Warunek początkowy: Użytkownik znajduje się w widoku menu

Scenariusz główny:

1. Użytkownik wybiera opcję z menu "Nowa rozgrywka".

2. Użytkownik wprowadza ścieżkę do istniejącego zestawu reguł.

3. Aplikacja wczytuje otrzymane dane i sprawdza poprawność reguł.

4. Aplikacja przekazuje przetworzone dane do wszystkich modułów (giełdy, tutoriala oraz map i rozgrywki)

5.Menu tworzy zlecenie wyświetlenia początkowej osady i przekazuje je do modułu Map i Rozgrywki

6.Aplikacja wyświetla widok mapy i rozgrywki

Scenariusz alternatywny:

2.1. W przypadku błędu otwarcia pliku użytkownik otrzymuje stosowny komunikat i propozycję wybrania innego pliku.

4.1. W przypadku niepoprawnej postaci zestawu reguł użytkownik otrzymuje stosowny komunikat i propozycję wybrania innego pliku

UC-10: Wyświetlenie informacji o imperium

Cel: Użytkownik chce wyświetlić informacje o imperium

Kontekst: Użytkownik dowiaduje się szczegółowych informacji o obecnym stanie rozgrywki, posiadanych domenach i zasobach, aby jak najlepiej je wykorzystać i rozplanować taktykę gry

Warunek początkowy: Użytkownik znajduje się widoku mapy i rozgrywki lub w widoku giełdy

Scenariusz główny:

1.Użytkownik zapoznaje się z graficzną reprezentacją ogólnych informacji o mieszkańcach, budynkach i zasobach.

2.W celu zapoznania się z informacjami szczegółowymi, użytkownik wybiera ikonę w interfejsie reprezentującą imperium.

3.Aplikacja wyświetla w odpowiednim panelu takie informacje, jak dane dot. budowli postawionych w imperium, przetwarzanych zasobów, liczbie mieszkańców o określonych kwalifikacjach, posiadanym kapitale, technologie dostępne w danym momencie.

UC-11: Przesunięcie widoku mapy

Cel: Użytkownik chce przesunąć widok mapy

Kontekst: Użytkownik uzyskuje dostęp do niewidocznych fragmentów mapy w celu ich podejrzenia lub zagospodarowania

Warunek początkowy: Użytkownik znajduje się w widoku mapy i kreatora

Scenariusz główny:

1. Użytkownik wciska i przytrzymuje LPM/przesuwa strzałkami

2. Aplikacja uaktualnia widok, pokazując żądany fragment mapy

Scenariusz alternatywny:

2.1. W przypadku dotarcia do granic imperium użytkownik zostaje o tym poinformowany i mapa nie zostaje przesunięta

UC-12: Postawienie budynku
Cel: Użytkownik chce postawić budynek
Kontekst: Użytkownik tworzy nowy budynek, w którym będzie korzystał z pewnych zasobów w celu stworzenia innych
Warunek początkowy: Użytkownik znajduje się w widoku mapy i kreatora
Scenariusz główny:
1.Użytkownik klika LPM na ikonę reprezentującą potrzebny budynek
2.Do kursora zostaje przyczepiony "widmowy budynek"
3.Użytkownik klika LPM w miejsce na mapie, w którym ma powstać budynek.
4.Budynek zostaje postawiony (umiejscowiony na mapie i zwizualizowany na radarze HUD) - od tej pory wpływa on na grę.
Scenariusz alternatywny:
2.1. W przypadku, gdyby obecny stan rozgrywki nie pozwalał na wykorzystanie danego budynku, użytkownik otrzymuje stosowną informację.
4.1. W przypadku wybrania miejsca na mapie nieodpowiedniego do postawienia budynku, następuje powrót do punktu 2. - użytkownik może spróbować postawić budynek w innym miejscu.

UC-13: Zburzenie budynku
Cel: Użytkownik chce zburzyć budynek
Kontekst: Użytkownik usuwa stary budynek, który nie jest niezbędny lub utrudnia rozwój domeny
Warunek początkowy: Użytkownik znajduje się w widoku mapy i kreatora
Scenariusz główny:
1.Użytkownik klika LPM na ikonę "Usuń"
2.Użytkownik najeżdża nad miejsce na mapie, w którym znajduje się budynek do wyburzenia i klika LPM.
3.Budynek zostaje usunięty (znika z mapy i z radaru HUD) - użytkownik odzyskuje część zainwestowanych zasobów.
Scenariusz alternatywny:
3.1. W przypadku wybrania miejsca na mapie, w którym nie ma żadnego budynku, użytkownik otrzymuje stosowny komunikat.

UC-14: Podgląd informacji o budynku

Cel: Użytkownik chce zobaczyć informacje o budynku
Kontekst: Użytkownik przypomina sobie dane budynku dot. wszystkich jego najważniejszych atrybutów.
Warunek początkowy: Użytkownik znajduje się w widoku mapy i kreatora
Scenariusz główny:
1.Użytkownik najeżdża kursorem na wybrany budynek.
2.Użytkownik odczytuje informacje z wyświetlonej przez aplikację notatki.

UC-15: Podgląd bilansu zasobów
Cel: Użytkownik chce zobaczyć bilans zasobów
Kontekst: Użytkownik dowiaduje się szczegółowych informacji o bilansie zasobów w danym momencie rozgrywki, aby jak najlepiej je wykorzystać i rozplanować taktykę gry
Warunek początkowy: Użytkownik znajduje się widoku mapy i rozgrywki lub w widoku giełdy
Scenariusz główny:
1.Użytkownik zapoznaje się z graficzną reprezentacją ogólnych informacji o zasobach.
2.W celu zapoznania się z informacjami szczegółowymi, użytkownik wybiera ikonę w interfejsie reprezentującą bilans zasobów.
3.Aplikacja wyświetla w odpowiednim panelu takie informacje, jak produkowane i konsumowane zasoby (wraz z szybkością produkcji i konsumpcji), ilość posiadanego kapitału, itp.

UC-16: Przełączanie między widokami
Cel: Użytkownik chce zobaczyć inny dostępny widok lub zakładkę
Kontekst: Użytkownik uzyskuje dostęp do niewyświetlanego obecnie widoku lub zakładki, celem uzyskania żądanych informacji lub wprowadzenia pewnych zmian.
Warunek początkowy: Użytkownik znajduje się w dowolnym z widoków dostępnych w grze.
Scenariusz główny:
1.Użytkownik zapoznaje się z opcjami dostępnymi w danym widoku.
2.Użytkownik klika LPM na ikonę wybranego widoku lub nazwę interesującej go zakładki

3.Użytkownik w nowym widoku/panelu zapoznaje się z wyświetlonymi informacjami i wprowadza zmiany.
4.Użytkownik wybiera "Zapisz".
5.Użytkownik znajduje się widoku, z którego wystartował w 1.
Scenariusz alternatywny:
3.1 W przypadku widoku tylko do odczytu użytkownik jedynie zapoznaje się z zaprezentowanymi informacjami.
3.1.1 W celu powrotu do poprzedniego widoku użytkownik klika LPM na przycisk "Zamknij".
4.1 W przypadku, kiedy użytkownik jeszcze nie otworzył widoku, na którym mu zależy, następuje powrót do 1. aż do momentu, kiedy użytkownik dotrze do opcji wyświetlenia widoku, który go naprawdę interesuje.

UC-17: Przejście do giełdy
Cel: Użytkownik chce zobaczyć giełdę
Kontekst: Użytkownik zapoznaje się z widokiem giełdy, aby zdecydować, w co chce zainwestować i ile kapitału może wykorzystać.
Warunek początkowy: Użytkownik znajduje się widoku mapy i rozgrywki
Scenariusz główny:
1.W panelu interfejsu użytkownik klika LPM na ikonę reprezentującą giełdę.
2.Rozgrywka zostaje zatrzymana, a użytkownik zapoznaje się z widokiem giełdy wyświetlonym w miejsce mapy.
3.Użytkownik klika LPM na interesujący go towar/papiery wartościowe.
4.Użytkownik otrzymuje informację o kursie akcji wybranego towaru.
Scenariusz alternatywny:
3.1 W przypadku chęci natychmiastowego powrotu do widoku mapy, użytkownik wybiera przycisk "Zapisz i wróć".
4.1 W przypadku chęci zapoznania się z innymi towarami, następuje powrót do 3.
4.2 W przypadku, gdy użytkownik decyduje się zagrać na giełdzie, następuje przejście do UC-18 pk1.

UC-18: Gra na giełdzie
Cel: Użytkownik chce zagrać na giełdzie
Kontekst: Użytkownik inwestuje swoje zasoby celem pomnożenia kapitału

Warunek początkowy: Użytkownik znajduje się widoku giełdy

Scenariusz główny:

1.Użytkownik klika LPM na interesujące go papiery wartościowe.

2.W menu giełdy użytkownik wybiera przycisk "Kup".

3.Użytkownik definiuje w polu tekstowym, ile kapitału chce zainwestować.

4.Użytkownik klika "Zapisz i wróć".

5.Wirtualny magazyn i zasoby gracza są aktualizowane, a użytkownik zostaje przeniesiony do widoku mapy i rozgrywki.

Scenariusz alternatywny:

2.1 W przypadku chęci sprzedaży użytkownik klika przycisk "Sprzedaj".

4.1 W przypadku, gdy użytkownik chce kontynuować granie na giełzie, następuje powrót do 1.

UC-19: Kupno towarów

Cel: Użytkownik chce zakupić towary

Kontekst: Użytkownik inwestuje swoje zasoby celem zdobycia potrzebnych mu towarów, ew. sprzedaje je, aby zwiększyć swój kapitał.

Warunek początkowy: Użytkownik znajduje się widoku giełdy

Scenariusz główny:

1.Użytkownik klika LPM na interesujący go towar.

2.W menu giełdy użytkownik wybiera przycisk "Kup".

3.Użytkownik definiuje w polu tekstowym, ile chce zakupić danego towaru.

4.Użytkownik klika "Zapisz i wróć".

5.Wirtualny magazyn i zasoby gracza są aktualizowane, a użytkownik zostaje przeniesiony do widoku mapy i rozgrywki.

Scenariusz alternatywny:

2.1 W przypadku chęci sprzedaży użytkownik klika przycisk "Sprzedaj".

UC-20: Zatrzymanie rozgrywki

Cel: Użytkownik chce zatrzymać rozgrywkę

Kontekst: Użytkownik chce zrobić przerwę w rozgrywce, zapisać ją lub całkowicie wyjść z aplikacji

Warunek początkowy: Użytkownik znajduje się widoku mapy i rozgrywki lub w widoku giełdy

Scenariusz główny:

1.Użytkownik wybiera ikonę zatrzymania rozgrywki.
2.Aplikacja wyświetla menu, w którym użytkownik może całkowicie opuścić rozgrywkę, zapisać rozgrywkę lub wyjść z aplikacji.

UC-21: Poznanie zasad gry
Cel: Użytkownik chce poznać zasady gry
Kontekst: Użytkownik chce skorzystać z tutoriala w celu zapoznania się z całością podręcznika dla użytkownika.
Warunek początkowy: Użytkownik jest w widoku menu.
Scenariusz główny:
1.Użytkownik wybiera opcję "Tutorial".
2.Użytkownik przegląda wyświetlaną pomoc do rozgrywki.
3.Użytkownik przemieszcza się między tematami za pomocą wyświetlanego z boku menu.

Bonus

Tu były gupie notatki Roberta, ale już ich nie ma MUAHAHAHAHHAHA