DOKUMENTACJA

- Opis projektu
- Opis klas
- Proponowana hierarchia
- Spis funkcji i opis roznych funkcjonalnosci

OPIS PROJEKTU

Projektem który tworzymy jest gra oparta na tradycyjnych „Chińczyku” (Ludo), dodatkowo dostępne są klasy postaci z różnymi umiejętnościami. Tutaj warto nadmienić, że wszystkie grafiki są wykonywane przez nas. Na tę chwilę proponowanych jest 6 klas postaci (Albali, Funi, Intan , Saph, Polaris, Samaya) a dodatkowo 3 (Lich, Venator, Liesma) będą przygotowane jeszcze w wesji Beta gry (ta liczba może się zwiększyć lub zmniejszyć, w zależności od stopnia skomlikowania, powtarzalności czy balansu (aby żadna z klas nie wyróżniała się na tle innych).

Poniżej próbka jak bedą wyglądać postacie (.gif):

https://imgur.com/q3CAcJL

oraz kość do gry:

https://imgur.com/sKRI3am

OPIS KLAS

- Launcher – klasa odpowiadająca za uruchomienie aplikacji
- Game – odpowiada za ustawienia okna (wielkość, tytuł, częstotliwość odświeżania,wątek), przechowuje też informacje o bierzącym stanie: Menu, Ustawienia, Preparation(stan przed właściwym uruchomieniem gry – wybór graczy, klas postaci), Gra
- Handler – przechowuje ważne informacje/zmienne aby mieć do nich łatwy dostęp w każdym miejscu

Wyświetlanie:

- Display – tworzy Jframe o zadanej wielkości oraz Canvas (na którym potem wyświetlane są grafiki), ustawienia okna

Stany gry:

- State (abstrakcyjna) – informacja o bierzącym stanie, getterry/settery obecnego stanu
- MenuState – dziedziczy po State, wybor pomiedzy grą a ustawieniami (w przyszłości inne funkcjonalności)
- SettingState – dziedziczy po State, ustawienia tj. szybkość animacji, częstotliwość odświeżania, rozmiar okna
- PrepState – dziedziczy po State, otrzymuje od graczy informacje na temat ilości Graczy i Botów w grze (później wybór „pustego" gracza, bez pionków)
- GameState – dziedziczy po State,

GFX:

- Assets – przechowuje wszystkie grafiki – pionki, mapa, przyciski, konstka
- ImageLoader – odpowiada za wczytywanie grafik
- SpriteSheet – przechowuje i przycina arkusz grafik wczytany w Assets do zadanych rozmiarów (np. Kostka do gry składa się z 6 podobnych części więc zostaje wczytana tylko jedna grafika a potem odpowiednio przycięta)

Input:

- MouseManager – pobiera aktualną pozycję kursora myszy po kliknięciu

Obiekty:

- Entity (abstrakcyjna) – pozycja na ekranie (x,y), szerokość wysokość
- Counter (abstrakcyjna) – dziedziczy po Entity, pionek – posiada informacje jak pozycję (x,y) początkową, pozycję startową, ilość klatek animacji (od niej zależy szybkość poruszania i długość animacji)

  Klasy postaci (dziedziczą po counter):

  - nazwy postaci zostały zaczerpnięte z nazw gwiazd (nazwy mają swoje znaczenie)

  - Albali – swallower - zbija wszystkie pionki które minie podczas trwania umiejetności specjalnej (1-2 tury)
  - Funi – fire / blaze - tworzy ogień na losowych polach mapy ok. 3-4 pola, (1-2 tury)
  - Intan – diamond - (passive) aby go zbić należy stanąć na jego pole dwa razy, za pierwszym razem zbijamy mu tarczę, za drugim dopiero zostaje przeniesiony do bazy gracza
  - Lich – undead creature controlling other undead creatures - wskrzesza zbity pionek na 3 tury, jeśli dany pionek dojdzie do pola końcowego to pozostaje na mapie, jeśli nie to nadal zostaje przeniesiony do bazy gracza
  - Polaris – polar star - Przesuwa pionek następnego gracza o wyrzuconą liczbę oczek (2-3 tury), inny pomysł: losowy gracz traci ruch (2 tury, gracz losowany co turę)
  - Samaya – peace - (passive) nie zbija przeciwników ale jej też nie da się zbić
  - Saph – sword of a gaint - wojownik
  - Venator – hunter - łowca
  - Liesma – fire – tworzy ogień na 2 miejscach przed sobą i za sobą na 3 rundy

- Player (abstrakcyjna) – posiada inforamcje jak: wybrane klasy gracza, punkty, kolor pionków, pozycja startowa, pozycja końcowa (na mapie)

  Typy graczy (dziedziczą po Player):
  - Person – aplikacja dostaje input od użytkownika
  - Bot – sam wybiera pionki w PreparationState oraz sam porusza się podczas gry

- Dice – dziedziczy po Entity, kostka do gry, posiada informacje takie jak: hitbox, ilosc wyrzuconych oczek, prawda/fałsz czy można w danym momencie rzucić kością + bardziej zaawansowana logika związana z graczami i ich miejscem na mapie np. jezeli dany gracz wyrzucił „szóstkę” 3 razy pod rząd to kostka przekazywana jest następneu graczowi. Kostka posiada animację rzutu (w wersji końcowej możliwość dostosowania w ustawieniach).

- Timer – dziedziczy po Entity, wyświetla obwódkę w kolorze gracza która upływa wraz z czasem- czas na ruch to 10 sekund (możliwe do dostosowania w ustawieniach

- Tile – jest to kafelek na mapie, posiada informacje takie jak: prawda/fałsz – czy można na danym „kafelku” zbijać innych graczy, pionki ktore znajduje sie na nim w danym momencie, pozycja na mapie

- Board – dziedziczy po Entity, posiada tablicę typu Tile, co za tym idzie przechowuje wszystkie „kafelki” na mapie

- UltimateBar – dziedziczy po Entity, wyświetla naładowanie umiejetności specjalnej (pasek w kolorze gracza)

(Ilość klas z pewnością się zwiększy)

SPIS FUNKCJI

(W podanym spisie nie uwzględniamy getterów i setterów)

init() – class: Game

- Konstruuje obiekt Display
- Inicjuje obiekt Assets
- Konstruuje Handler
- Konstruuje stany gry i ustawia bieżący

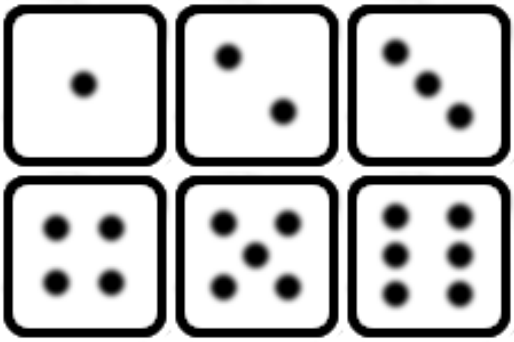
tick() – class: Game, Counter,Dice (i inne wymagające odświeżania) wykonywana wiele razy na sekundę (standardowo 60)– aktualizuje pozycje graczy co kilka px

render()-również wykonywana wiele razy na sekundę - aktualizuje grafikę graczy co wraz ze zmienioną pozycją daje wrażenie animacji

init() – class: Assets, wczytuje wszystkie grafiki, aby nie były wczytywane przy każdym ich wywołaniu

getInput – class: Person – pobiera od gracza input w postaci wyboru pionka

crop() – class: SpriteSheet, zwraca wycięty obrazek z siatki np. z siatki roll.png (225x150px). (poniżej)



zwraca oraz dodaje png do tablicy obrazów typu BufferedImage (75x75px). (przykład poniżej)

OPIS FUNKCJONALNOŚCI

Po wejściu do głównego menu będzie można wybrać grę lub ustawienia. Ustawienia to m.in. szybkość animacji, wielkość ekranu. W planie jest możliwość włączenia wywoływania umiejętności specjalnej poo naciśnięcciu odpowiedniego klawisza na klawiaturze. Po wejściu do gry pojawia się ekran z wyborem graczy i/lub botów oraz możliwością zmiany nicków. Kolejne ekrany to wybór klas postaci dla każdego gracza, gracze mają określony czas na wybór w przeciwnym wypadku postacie zostaną wybrane losowo. Kolejnym etap jest właściwa gra. Gracz rozpoczynający rozgrywkę jest wybierany losowo. Po rzucie kostką musi wybrać postać którą chce się poruszyć. Gdy jego umiejętność specjalna zostaje naładowana może jej użyć w swojej turze (przed rzutem kostką). Gra kończy się gdy wszyscy gracze umieszczą swoje pionki na pozycji końcowej. Na koniec zostaje wyświetlona tabela, wygrywa gracz który pierwszy umieści pionki na pozycji końcowej.