PROGRAMOWANIE OBIEKTOWE - PROJEKT

DOKUMENTACJA KOŃCOWA

Ludo Legends

Aplikacja w języku Java

Informacje

Kod kursu: **INEW00003P**

Nr grupy zajęciowej: **E06-42g**

Autorzy:

Krystian Ogonowski

Antoni Filipkiewicz

Wprowadzenie

Projetem który realizujemy jest gra oparta na tradycyjnym „Chińczyku" (Ludo), dodatkowo dostępne są różne umiejętności specjalne cechujące każdą z klas. Grafiki są autorskie. Zaimplementowane klasy postaci to: Albali, Altair, Funi, Intan, Mira, Polaris Samaya, Saph.

Spis treści

Opis klas

Package ludogame

Launcher – klasa odpowiadająca za uruchomienie aplikacji

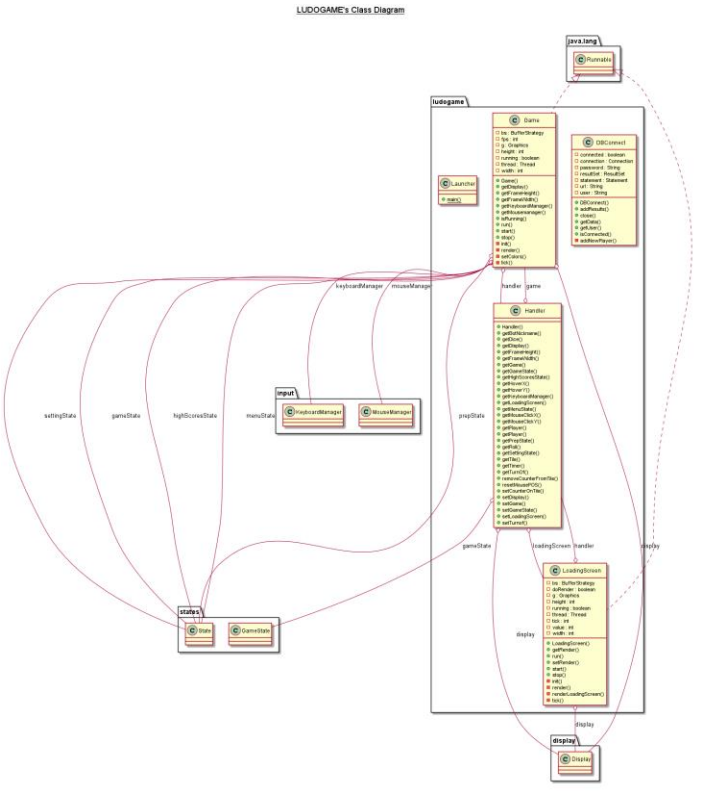
Game – odpowiada za ustawienia okna (wielkość, tytuł), przechowuje też informacje o bierzącym statusie aplikacji (menu, ustawienia, tabela wyników, gra)

LoadingScreen – odpowiada za wszelkie ekrany ładowania m.in. podczas wczytywania aplikacji oraz podczas łączenia z bazą danych

DBConnect – łączy aplikację z bazą danych, pobiera i zapisuje dane graczy

Handler – odpowiada za komunikację pomiędzy klasami, składa się w znacznej większości z getterów i setterów

Error - odpowiada za wyświetlanie komunikatów np. o błędnym wyborze pionków



Package states

State – wybór aktualnego stanu aplikacji

MenuState – ekran początkowy, stąd gracz przechodzi do gry, ustawień, tabeli wyników

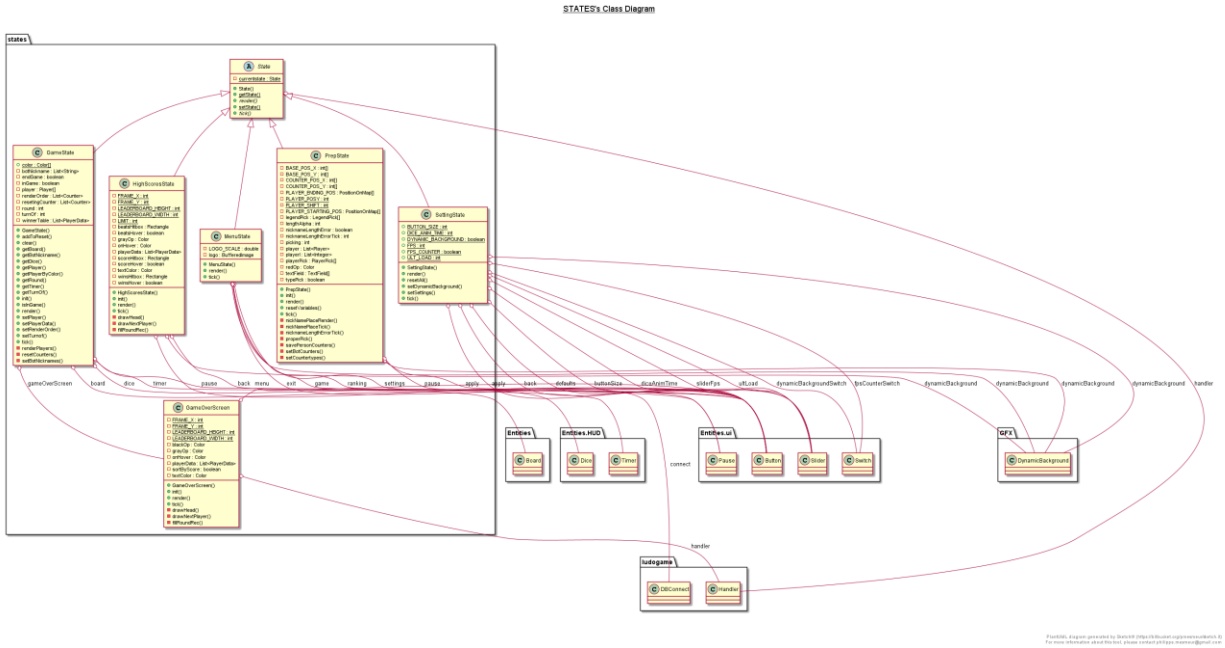
SettingState – przechowuje ustawienia gry, niektóre z nich mogą być zmienione przez użytkownika

HighScoresState – pokazuje dane graczy znajdujących się w bazie MySQL

PrepState – Pobiera od użytkownika dane o grze tj. typ gracza (gracz, bot) oraz typ używanych klas postaci

GameState – odpowiada m.in. za ruch odpowiedniego gracza oraz ustala kolejność renderowania pionków

GameOverScreen – wyświetla ranking graczy biorących udział w grze, przekazuje dane do zapisania w bazie danych



Package players

PlayerData – przechowuje dane gracza tj. ID, nick, zdobyte punkty,zbicia

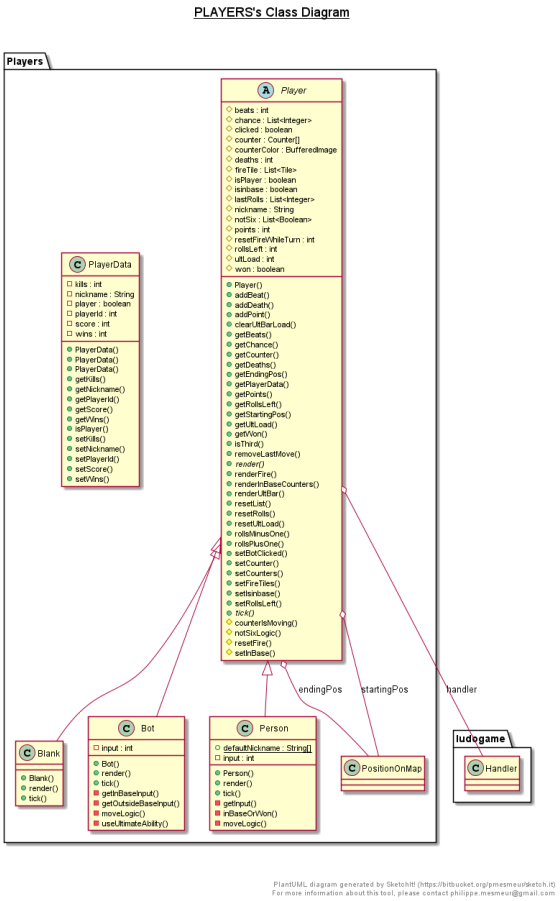
PositionOnMap – przechowuje aktualną pozycję gracza na mapie (indeksy dynamicznej tablicy dwuwymiarowej)

Player – posiada dane o graczu: wybrane pionki, wyniki rzutów kością w turze, pozycje (końcową i początkową) kolor pionków itp.

Person – tzw. Użykownik który ma kontrolę nad grą

Bot – klasa wykonująca tury bez ingerencji użytkownika

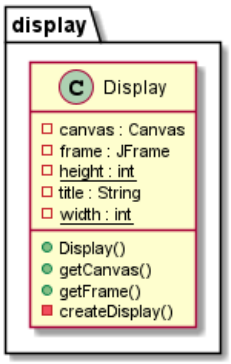
Blank – gracz bez pionków i możliwości poruszania, nie wpływa na grę



Package display

Display – tworzy okno (Jframe) oraz obsługuje jego ustawienia tj. : pozycja na ekranie, wielkość

DISPLAY's Class Diagram



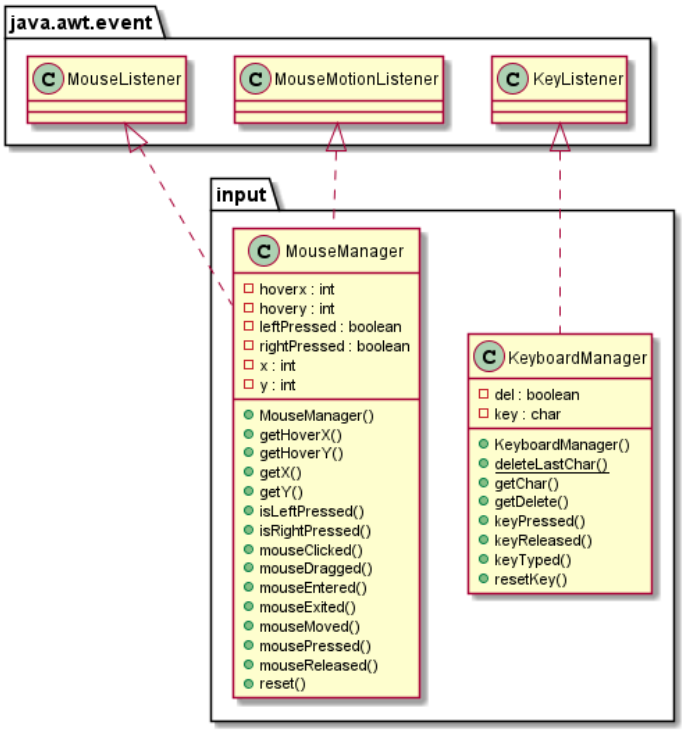
PlantUML diagram generated by SketchIt! (https://bitbucket.org/pmesmeur/sketch.it)
For more information about this tool, please contact philippe.mesmeur@gmail.com

Package input

MouseManager – implementuje interfejsy z java.awt.event: MouseListener, MouseMotionListener i MouseWheelListener, odpowiedzalna za odbieranie odpowiednich informacji o ruchu myszy, kliknięciu przycisków czy ruchu kółkiem myszy

KeyboardManager -implementuje interfejs Keylistener, odpowiedzialna za odbieranie informacji o naciśnięciu klawiszy przez użytkownika

INPUT's Class Diagram



PlantUML diagram generated by SketchIt! (https://bitbucket.org/pmesmeur/sketch.it)
For more information about this tool, please contact philippe.mesmeur@gmail.com

Package GFXandEffects

Assets – klasa wczytująca i przechowująca grafiki do gry

Description – przechowująca opisy postaci

DynamicBackground – odpowiada za poruszanie i wyświetlanie animowanego tła w menu

SoundEffect – odtwarza dźwięki

SpriteSheet – wczytuje i przechowuje siatkę grafik tj. kostki do gry, pionków czy przycisków

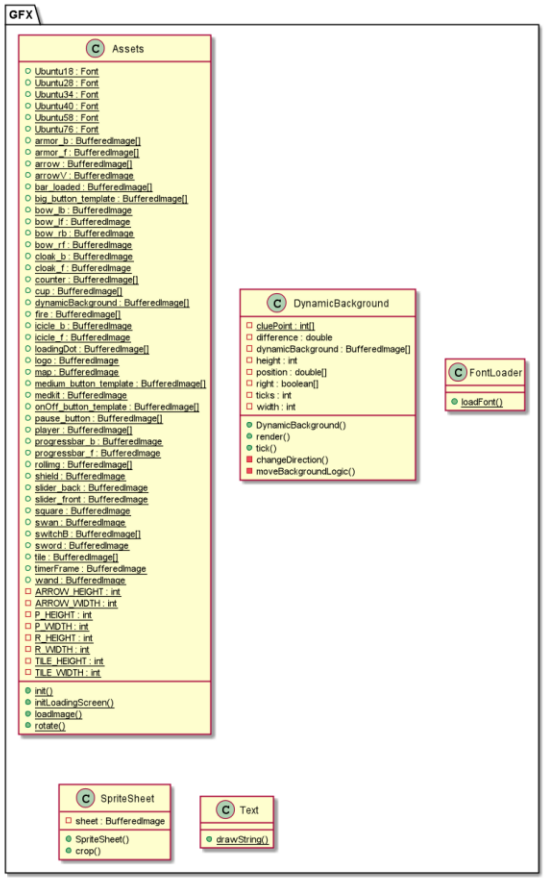
FontLoader – posiada metodę statyczną wczytującą dany font z pliku

Text – wypisuje na ekranie text

Przykładowe siatki tekstur:

Przykładowe grafiki specjalne postaci:

GFX's Class Diagram



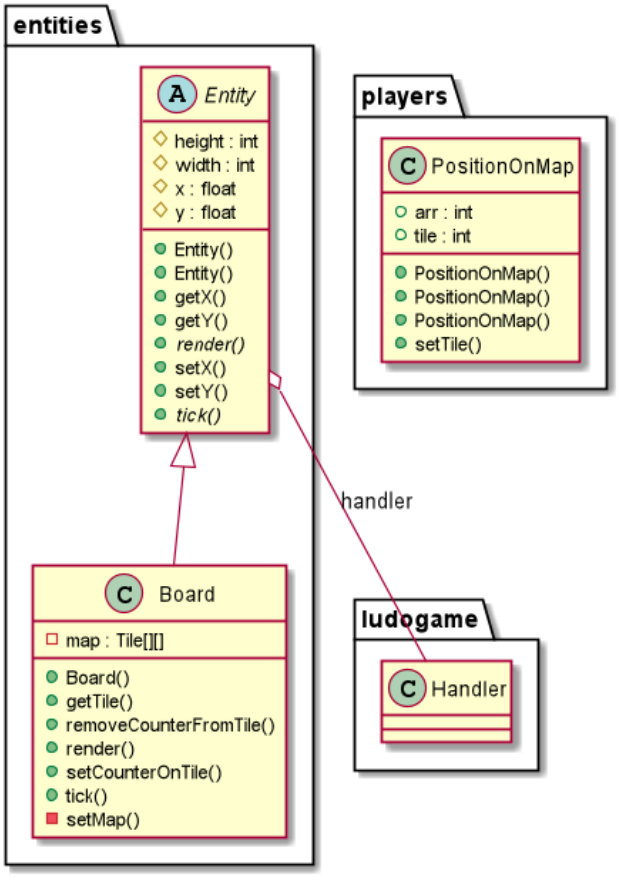
Package entities

Zawiera: package counters, HUD i ui

Board – przechowuje dynamiczną dwuwymiarową tablicę pól

Entity – abstrakcyjna – posiada pozycję elementu na ekranie oraz jego wysokość i szerokość

ENTITIES's Class Diagram



PlantUML diagram generated by SketchIt! (https://bitbucket.org/pmesmeur/sketch.it)
For more information about this tool, please contact philippe.mesmeur@gmail.com

Package entities.counters

Counter – przechowuje informacje o pozycji pionka na mapie, o tym czy w danym momencie się porusza lub wraca do ‘bazy’, informacje o umiejętnościach, ilości pól które przebył

Klasy dziedziczące po ‘Counter’ – klasy postaci

Aggitarius – (niezaimplementowana) – umiejętności nieznane

Albali – posiada pelerynę, po użyciu umiejętności staje się niewidzialny, nie można go wtedy zbić oraz jest odporny na ogień

Altair – posiada skrzydła, co ruch dolosowuje się liczba 1-2 która zwiększa ilość pól do przebycia

Funi – posiada laskę, po użyciu umiejętności generuje ogień na dwóch losowych polach, który istnieje przez kilka rund

Intan – posiada tarczę, po stanięciu na jego pole tarcza się niszczy, przy kolejnym stanięciu pionek zostaje zbity

Lich – (niezaimplementowana) – umiejętności nieznane

Mira – posiada apteczkę – może przywrócić losowy pionek który był wcześniej zbity

Polaris – nosi pancerz - aby został zbity trzeba stanąć na jego pole aż trzy razy, jego minusem jest powolne poruszanie

Samaya – łabędź symbolizuje pokój, jej umiejętnością pasywną jest to, że nie może zostać zbita, ale też nie może zbijać innych przeciwników

Saph – posiada miecz, po użyciu umiejętności specjalnej zbija wszystkie pionki przeciwników które napotka

Package entities.HUD

BarParticle – adpowiada za grafikę i animację przy naładowaniu umiejętności specjalnej

Dice – kość do gry, losuje liczbę, dodatkowo gdy gracz wystarczająco długo nie wyrzuci 6 oraz wszystkie jego pionki znajdują się w ‘bazie’ zwiększa szansę na wyrzucenie 6

Timer - odlicza czas po którym kość zostaje automatycznie rzucona lub odpowiednio - wybrany pionek którym ma poruszyć się gracz

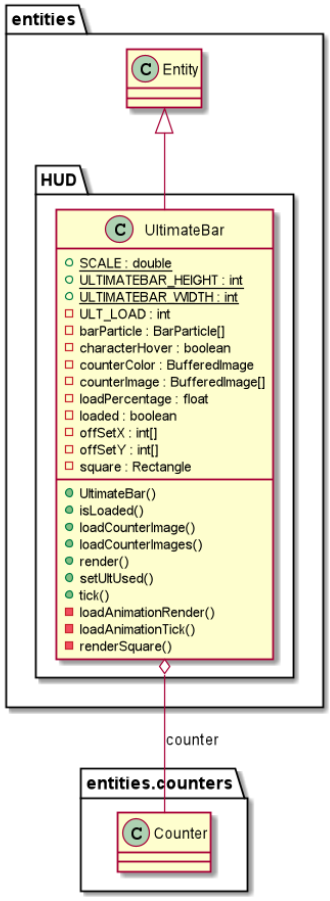
UltimateBar – wyświetla pasek naładowania umiejętności, posiada też kafelek którym używa się umiejętności dla danego pionka

Przykładowe grafiki

- **kości i timer’a**

- **ultimate bar’u**

HUD's Class Diagram



PlantUML diagram generated by Sketchit! (https://bitbucket.org/pmesmeur/sketch.it)
For more information about this tool, please contact philippe.mesmeur@gmail.com

Package entities.ui

Button – przycisk, pojawia się m.in. w menu i ustawieniach

LegendPick – okno wyboru postaci, sklada się miedzy innymi z CounterTiles

CounterTile – wyświetla kafelek do wyboru postaci

Info – wyświetla opisy i podpowiiedzi na temat klas postaci

InfoTile – kafelek w info posiada grafikę klasy, opis oraz przedstawia jej plusy i minusy

Pause – pauza, zatrzymuje grę, można z niej wyjść do menu

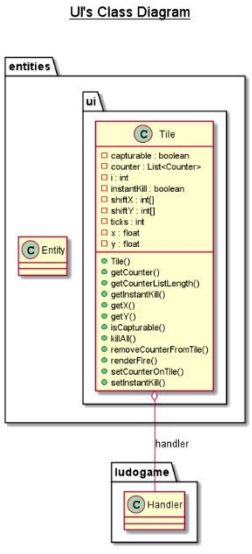
PlayerPick – jest to okno wyboru typu graczy, posiada też pola tekstowe do wpisania 'nick'ów'

Slider – pozwala zmieniać ustawienia tj. czas na rzut kością podczas gry – jest przedziałem liczbowym

Switch – pozwala zmieniać ustawienia tj. obecność animowanego tła w menu – reprentuje wartości 0/1

TextField – pozwala na wpisywanie nick'ów graczy

Tile – przechowuje informacje o pionkach na danym polu, odpowiada też za logikę skalowanie pionków



Przykładowe grafiki

- **przycisków**

- **pola tekstowego**

Karty CRC

Package ludogame

Abstract **Launcher**	
• Przechowuje metodę main • Włącza program	• Game • Handler • LoadingScreen

LoadingScreen	
• Wyświetlanie ekranu ładowania	• Handler • Display

DBConnect	
• Połączenie z bazą danych	• PlayerData

Handler	
• przechowywanie dostępów do metod i zmiennych globalnych	• Game

Game	
• obsługuje jeden z dwóch wątków aplikacji • inicjuje stany aplikacji	• Display • Handler • Connect • MouseManager • KeyboardManager

Package states

Abstract **State** GameState, MenuState, PrepState, SettingState, HighScoreState	
• Podział programu na stany (menu, gra, opcje itd.)	• Game

State **SettingState**	
• Umożliwienie użytkownikowi ustawienie kilku parametrów	• Slider • Button • DynamicBackground • Switch

State **MenuState**	
• Wyświetlenie menu programu	• Button • DynamicBackground • Assets

State **HighScoreState**	
• Wyświetlenie rankingu graczy	• PlayerData • DBConnect • Button • DynamicBackground • Text • Assets

State	
PrepState	
• Wybór graczy oraz botów • Nazwanie graczy • Wybór pionków	• Button • LegendPick • DynamicBackground • Pause • Info • TextField • Player • Text • PositionOnMap

State	
GameState	
• Wyświetlenie właściwej gry • Przeprowadzanie tur • Nadanie imion botom	• Player • Board • Dice • Timer • Pause • GameOverScreen • Counter • Tile • PositionOnMap

GameOverScreen	
• Wyświetlenie ekranu końcowego	• PlayerData • Text • Button • Assets

Package players

Abstract **Player** Blank, Bot, Person	
• Przechowywanie informacji gracza (kolor, pionki, punkty)	• Counter • PositionOnMap • Tile

PlayerData	
• Przechowywanie punktów graczy	

Player **Blank**	
• Stworzenie braku gracza	• PositionOnMap

PositionOnMap	
• Przechowywanie informacji odnośnie pozycji pionka	

Player **Person**	
• Stworzenie gracza-użytkownika • Logika ruchu i umiejętności specjalnej	• Counter • PositionOnMap • Assets • Text

Player **Bot**	
• Stworzenie gracza-bota • Logika ruchu i umiejętności specjalnej	• Counter • PositionOnMap • Assets • Text

Package input

MouseManager	
• Obsługa myszy	

KeyboardManager	
• Obsługa klawiatury	

Package GFXandEffects

Assets	
• Wczytywanie obrazów z plików do programu • przechowywanie obrazów oraz czcionek	

Description	
• przechowywanie tesktów	

DynamicBackground	
• Wyświetlanie dynamicznego tła	

FontLoader	
• Wczytywanie czcionek z pliku do programu	

SpriteSheet	
• Wczytywanie siatek tekstur	• Assets

Text	
• Rysowanie tekstu	

Package entities

Abstract **Entity** Board, Button, CounterTile, Info, InfoTile, LegendPick, Pause, PlayerPick, Slider, Switch, TextField, UltimateBar, BarParticle, Dice, Timer, Counter	
• Tworzenie obiektów posiadających położenie na ekranie	

LegendPick	Entity
• Stworzenie tabeli do wybierania pionków	• Assets

Display	
• Stworzenie okna programu	

InfoTile	Entity
• Stworzenie pola z informacjami do ekranu informacyjnego	• Assets • Text • Description • Albali • Funi • Intan • Mira • Polaris • Samaya • Saph • Aggitarius • Altair

Package display

PlayerPick	Entity
• Obsługa wyboru graczy	• Assets

Info	Entity
• Obsługa ekranu informacyjnego	• InfoTile

Button	Entity
• Obsługa przycisków	

Slider	Entity
• Obsługa suwaków	• Assets • Text

Tile	
• Przechowuje pionki graczy na polach • Logika stawania pionka na pole • Renderuje grafiki specjalne pól takie jak ogień • Odpowiedzialna za skalowanie	• Assets • Counter

Switch	Entity
• Obsługa przycisków typu ON/OFF	• Assets • Text

TextField	Entity
• Stworzenie pola tekstowego na nazwę gracza	• Assets • Text • KeyboardManager

UltimateBar	Entity
• Obsługa pasku ładowania ultimate	• Counter • Assets

BarParticle	Entity
• Stworzenie efektów wizualnych na ultimatebarze	• Assets

Dice	Entity
• Obsługa rzutu kością	• Assets

Timer	Entity
• Obsługa odliczania czasu	• Assets

Abstract	Entity
Counter Aggitarius, Albali, Altair, Funi, Intan, Lich, Mira, Polaris, Samaya, Saph	
• Uniwersalna logika ruchu pionka • Logika zbicia pionka	• UltimateBar • Handler • PositionOnMap • Dice • Timer • GameState • Tile

Albali	Counter
• Przechowywanie logiki umiejętności oraz wyglądu pionka Albali	

Funi	Counter
• Przechowywanie logiki umiejętności oraz wyglądu pionka Funi	

Intan	Counter
• Przechowywanie logiki umiejętności oraz wyglądu pionka Intan	

Lich	Counter
• Przechowywanie logiki umiejętności oraz wyglądu pionka Lich	

Mira	Counter
• Przechowywanie logiki umiejętności oraz wyglądu pionka Mira	

Polaris	Counter
• Przechowywanie logiki umiejętności oraz wyglądu pionka Polaris	

Samaya	Counter
• Przechowywanie logiki umiejętności oraz wyglądu pionka Samaya	

Saph	Counter
• Przechowywanie logiki umiejętności oraz wyglądu pionka Saph	

Altair	Counter
• Przechowywanie logiki umiejętności oraz wyglądu pionka Altair	

CounterTile	Entity
• Stworzenie obrazu pionków	• Assets

Aggitarius	Counter
• Przechowywanie logiki umiejętności oraz wyglądu pionka Aggitarius	

Pause	Entity
• Obsługa ekranu pauzy	• Assets • Button