**Point**

Reprezentuje punkt i vector 2D.

**Pola**

x – x współrzędna.

y – y współrzędna.

**Konstruktor**

Point(int x, int y) – konstruje nowy punkt x, y.

**static MapManager**

MapManger używany do tworzenia poziomu przy pomocy clasy Cell, dystrybucja przedmiotów i wrogów, wyświetlanie mapy.

**Pola**

h – Wysokość mapy (Tylko do odczytu).

w – Szerokość mapy (Tylko do odczytu).

h – Wysokość komórki (Tylko do odczytu).

w – Szerokość komórki (Tylko do odczytu).

map – Tablica wszystkich obiektów na mapie.

cellMap – Tablica wszystkich komórek.

**Metody**

void CreateBase() – Tworze pusty poziom.

Cell connectNearestRoom(Cell currentCell) - Lącze pokój w danej komórce <currentCell> z najbliższym pokojem po prawej stronie.

void GenerateDungeon() - Generuje poziom.

void DrawAllMap() - Wypisuje cały poziom.

bool IsVisible(Point objPos, Point pPos) - Zwraca True jeśli gracz w punkcie <pPos> widzi pewien objekt w punkcie <objPos>.

void FOV(Point pos) - Wyświetlanie mapy z perspektywy gracza. <pos> pozycja gracza.

void Decorate() - Dodaje wrogów i przedmioty do wygenerowanego poziomu.

void LootSpawner() - Dodaje przedmioty do wygenerowanego poziomu.

Item RandomItem() – Zwraca losowy przedmiot

void EnemySpawner() - Dodaje wrogów do wygenerowanego poziomu.

Enemy RandomEnemy() – Zwraca losowego wroga.

**Cell**

Reprezentuje podział mapy na komórki które zawierają pokoje.

Używany do tworzenie pokoi i korytarze.

**Pola**

room - Tablica obiektów ktore tworzą pokój (Tylko do odczytu).

posCell – Index w tablice MapManager.cellMap (Tylko do odczytu).

pivot - Index w tablice MapManager.map (Tylko do odczytu).

h – Wysokość pokoju w komórce.

w – Szerokość pokoju w komórce.

inConnection – Korytarz wchodzący do pokoju.

outConnection - Korytarz wchodzący do pokoju.

tickets –

**Metody**

Point RoomRandomPoint – Zwraca losowy punkt w pokoju.

bool cellIsEmpy(Point posCell) – Zwraca True jeśli komórka <posCell> jest pusta.

void AddCell() – Wpisuje dane o pokoju do MapManager.map.

void AddCorridor(Object[,] corridor, int len, int w, Point start) – Wpisuje dane z parametru <corridor> do MapManager.map.

Point SetPivot(Point posCell, int hRoom, int wRoom) – Ustawia pivor pokoju <posCell> względem jego wysokości <hRoom> i szerokości <wRoom>. Zwraca pivot pokuju.

void GenerateRoom(Point posCell) – Generuje pokuj w komurce <posCell>.

void GenerateSimpleCorridor(bool horizonatal, Point start, int len) – Generuje korytarz pewnej długości. Parametry: <horizontal> czy jest poziomy, <start> punkt na płaszczyżnie gdzie będzie generowany, <len> długość

void GenerateLtypeCorridor(bool down, bool left, int len, int height, Point start) – Generuje L-type corridor. Parametry: <down> czy najpierw generuje korytarz do dołu, <left> True jeśli generuje korytarz na lewo, <len> długość, <height> wysokość, <start> punkt na płaszczyżnie gdzie będzie generowany.

void GenerateZtypeCorridor(bool horizontal, bool up, int len, int height, Point start) –

Generuje korytarz typu Z. Parametry: <horizontal> czy lącze pokoje na tej samej linii, <up> True jeśli idzie do góry, <len> długość, <height> wysokość, <start> punkt na płaszczyżnie gdzie będzie generowany.

void ConnectCells(Cell c1, Cell c2, bool notDown = false) - Lączy dwa pokoje <c1> i <c2>. zakladamy że c1 jest wyżej i po lewej od c2. <notDown>Czy jest zakazane polączenie przez dól.

**Konstruktor**

Cell(Object [,] room, Point posCell, Point pivot, int h, int w)

**Object**

Reprezentuje obiekt na mapie.

**Pola**

impassable – Czy obiekt jest nieprzekraczalny.

image – Reprezentacja obiektu w grafice ASII.

color – Kolor obiektu.

visible – Czy obiekt był wypisany przynajmniej raz.

**Konstruktor**

Object(bool impassable, char image, ConsoleColor color)

**Door** : **Object**

Reprezentuje obiekt drzwi.

**Metody**

void OnAction() – reakcja na działanie. Otwieranie, zamykanie drzwi.

**Pola**

isOpen – Czy są otwarta.

**Konstruktor**

Door() : base(true, 'D', ConsoleColor.DarkYellow)

**Exit : Object**

Wejście do następnego poziomu.

**Metody**

void OnAction() – reakcja na działanie. Przejście na następny poziom

**Konstruktor**

Exit() : base(false, '>', ConsoleColor.DarkCyan)

**Interface**

Zawiera metody do wypisywania interfejsu i stale.

**Metody**

void drawContainers() – Wypisuje ramki interfejsu.

**Pola**

interfaceLen – długość interfejsu (Tylko do odczytu).

interfaceHight – wysokość interfejsu (Tylko do odczytu).

dopInterfaceLen – długość dodatkowego interfejsu (Tylko do odczytu).

dopInterfaceHight – wysokość dodatkowego interfejsu (Tylko do odczytu).

outputLine – y współrzędna pozycji kursora dla wypisywania tekstu w interfejsie.

outputRow – x współrzędna pozycji kursora dla wypisywania tekstu w interfejsie.

dopOutputLine – y współrzędna pozycji kursora dla wypisywania tekstu w dodatkowym interfejsie.

dopOutputRow – x współrzędna pozycji kursora dla wypisywania tekstu w dodatkowym interfejsie.

dopInterfaceIsEmpty – Czy dodatkowy interaface jest pusty.

**Player**

Reprezetnuje gracza. Zawiera wszystkie statystykie gracza i metody do kontrolowania.

**Pola**

x – x współrzędna pozgrycji gracza.

y – y współrzędna pozycji gracza.

prevTile – Inny obiekt który jest w tym samym punkcie co gracz.

exp - punkty doświadczenia (Tylko do odczytu).

lvl – poziom (Tylko do odczytu).

nextLevelExp – ilość punktów doświadczenia potrzebnych do zdobycia następnego poziomu (Tylko do odczytu).

perks – perks points (Tylko do odczytu).

str – siła (Tylko do odczytu).

basicCons – bazowa konstytucja gracza (Tylko do odczytu).

cons - obecna konstytucja gracza (Tylko do odczytu).

maxHp – maksymalna ilość Hp.

currentHp – obecna ilość Hp (Tylko do odczytu).

alive – pokazuje czy gracz jest żywy.

doneTurn - pokazuje czy gracz zakończył swoją ture.

inventory – tablica przedmiotów gracza.

equipped - wyposażone przedmioty.

armor – wskaźnik zbroi. (Tylko do odczytu).

**Metody**

void Deploy(int x, int y) - Rozmieścia objekt w punkcie x, y.

void Deploy() - Rozmieścia objekt.

void addExp(int amount) - Dodaje exp. <amount> Ilość punktów.

void LvlUp() - Zwieksza poziom gracza.

void ChangeHp(int amount, bool ignoreArmor = false) - Dodaje <amount> do player.currentHp. <ignoreArmor> czy ignoruje wskażnik zbroi.

void Death() - Śmierć gracza. Wywoluje koniec sesji.

void updateStats() - Aktualizuje statystyki gracza.

Attack(Point dir) - Wywoluje ChangeHp() na sąsiednim obiektie w kierunku <dir>

void EquipItem(int id) - Dodaje do pola equiped obiekt z inwentarza o indeksie <id>.

void ConsumeItem(int id) - Wywoluje Effect() przedmiotu z inwentarza.

Point GetDirection(ConsoleKey input) - Konwertuj dane wejściowe <input> z numpad na kierunek.

void GetAction(ConsoleKey input) - Sprawdza wejściowe dane <input> i wywoluje odpowidnie polecenie.

void Controller() - Uzyskuje znak wciśnięty przez uzera i wywoluje odpowidnie polecenie.

void writePlayerStatus() - Wypisuje statystyki gracza.

**Enemy**

Zawiera wszystkie metody kontrola AI i parametry wrogów.

**Pola**

name – imię.

attackPow –siła ataku.

radius – promień, w którym widzi gracza (Tylko do odczytu).

maxHp – maksymalna ilość Hp.

currentHp – obecna ilość Hp (Tylko do odczytu).

alive – pokazuje czy gracz jest żywy.

x – x współrzędna pozgrycji gracza.

y – y współrzędna pozycji gracza.

prevTile – Inny obiekt który jest w tym samym punkcie co gracz.

exp - punkty doświadczenia.

**Metody**

void Deploy(int x, int y) - Rozmieścia objekt w punkcie x, y.

bool InRange(Point point) - Zwraca True jeśli punkt <point> jest w promieniu.

Point CheckPlayer(int radius) – prawdza czy gracz jest w promieniu widzenia. Zwraca True jeśli gracz jest w promieniu.

void ChangeHp(int amount, bool ignoreArmor = false) - Dodaje <amount> do player.currentHp. <ignoreArmor> czy ignoruje wskażnik zbroi.

void Death() - Dodaje exp graczu. Ustawia że wrog jest martwy.

void MoveTo(Point target) - Przesuwa obiekt do celu.

void Attack(Point dir) - Wywoluje ChangeHp() na obiektie w zadanym kierunku.

void AttackPos(Point pos) - Wywoluje ChangeHp() na obiektie w zadanym punkcie.

**Konstruktor**

Enemy(char image, ConsoleColor color, int attack, int radius, int hp, int exp) : base(true, image, color)

**Bat :** **Enemy**

Reprezentuje wroga ‘Bat’. Ma zastąpiony sposób poruszania.

**Metody**

void MoveTo(Point target) - Przesuwa obiekt do <target>, ale tylko po przekątnej.

**Slime : Enemy**

Reprezentuje wroga ‘Slime. Po śmierci tworze jeście dwóch wrogów.

**Metody**

void Death() - Dodaje exp graczu. Ustawia że wrog jest martwy. Jeśli rek == True to tworzy jeście co najwyżej 2 Slime.

**Pola**

rek – True jeśli tworze wrogów śmierci.

**Item : Object**

Klas ogólny dla przedmiotów których gracz może użyć. Jest klasem bazowym dla Wearable i Consumable.

**Pola**

name – imię.

description – opis.

level – poziom przedmiotu.

**Konstruktor**

Item(string name, char image, ConsoleColor color, int level, string description) : base(false, image, color)

**Wearable : Item**

Klas dla przedmiotu, który gracz może wyposażyć. Zawiera metode do opisu przedmiotu.

**Metody**

void ShowInfo() - Pokazuje informacje o przedmiocie.

**Pola**

armor - wpływa na wskaźnik zbroi przy wyposażeniu.

damage - wpływa na siłe ataku przy wyposażeniu.

str – wpływa na wskażnik siły.

cons - wpływa na wskażnik konstytucji.

**Consumable: Item**

Klas dla przedmiotu, który gracz może zużyć. Zawiera metode do opisu przedmiot i efekt przedmiotu.

**Metody**

void ShowInfo() - Pokazuje informacje o przedmiocie

virtual void Effect(Player player) – metoda virtualna.

**static SessionManager**

Zawiera metody do urochomienia gry i podalszego działania.

**Metody**

void StartSession() – Urochomia grę.

void NextLevel() - Tworzy następny poziom.

void Update()

**Pola**

enemies – wszystkie aktywne przecziwniki na mapie.

player – obiekt gracza.

level – numer poziomu.