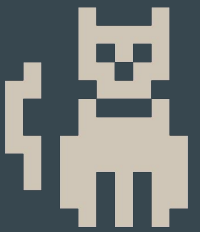
```
namespace roguelike
{
    title Cat's Minion

    {
        ProjectManager Jarosław Kaszczak;
        Developer Dominika Barrett;
        Developer Krzysztof Fieber;

    }
}
```



Project Overview

Create Roguelike game

- Tile-Based
- Real-time interaction between characters
- Labyrinth exploration
- When Player dies game restarts
- Levels contain collectable items, fights with enemies, finding treasure
- Saving state of the game
- Tests of the game

GameStory =

“In front of the gate Character meets a Cat.

Creature gives a Player the key and send for a mission.

Player opens gate and explores cellar.”

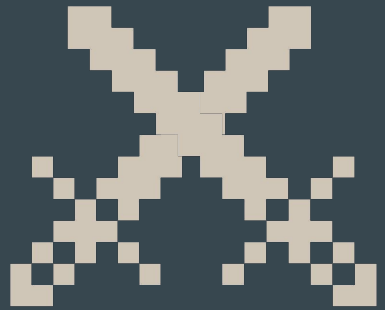
Enemies: **Bats**, **Ghost**

Collectable items: **Bone**, **Coin**, **Armor**, **Weapons**, **Helmets**, **Food**

UsedTools;

<Inspiration>

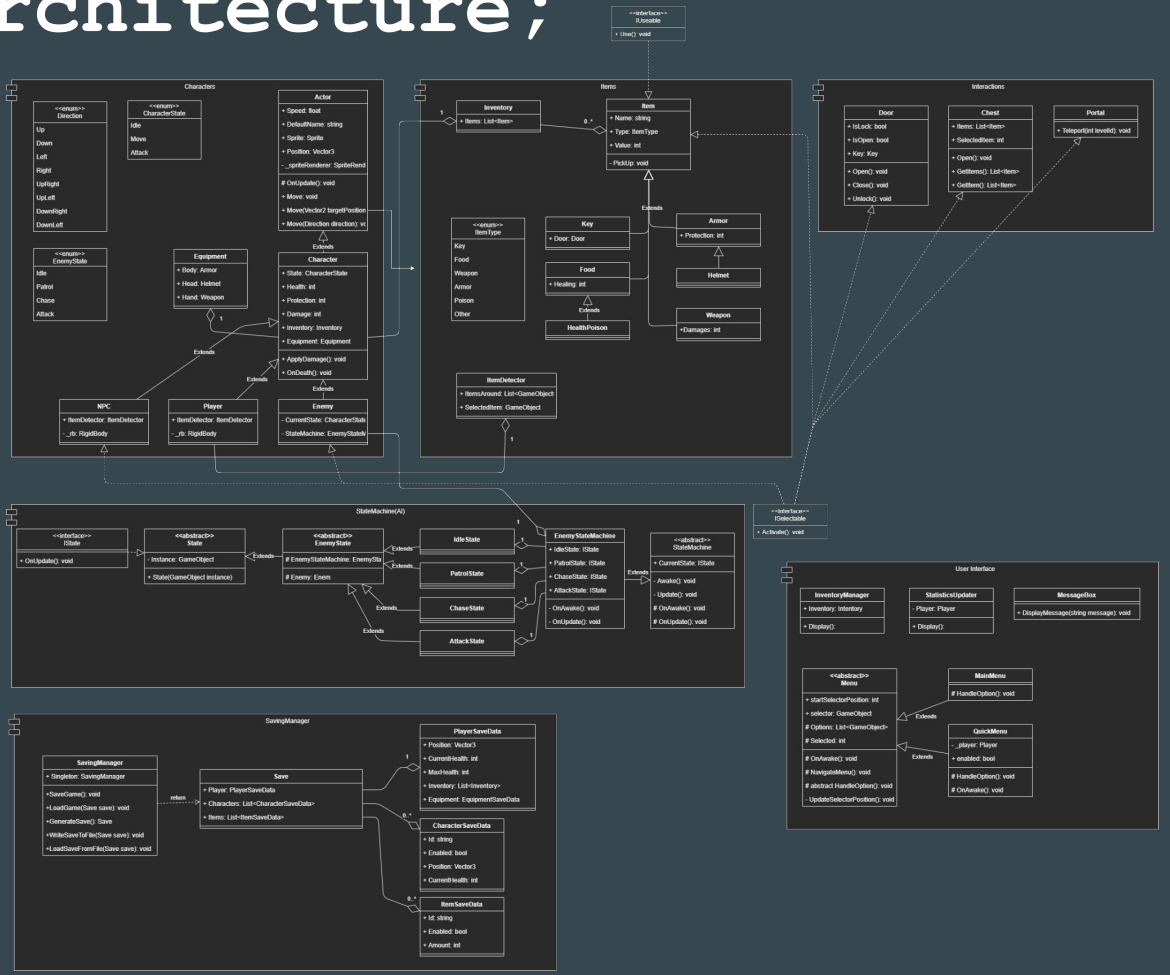
Let’s play!



using **Game.Architecture;**

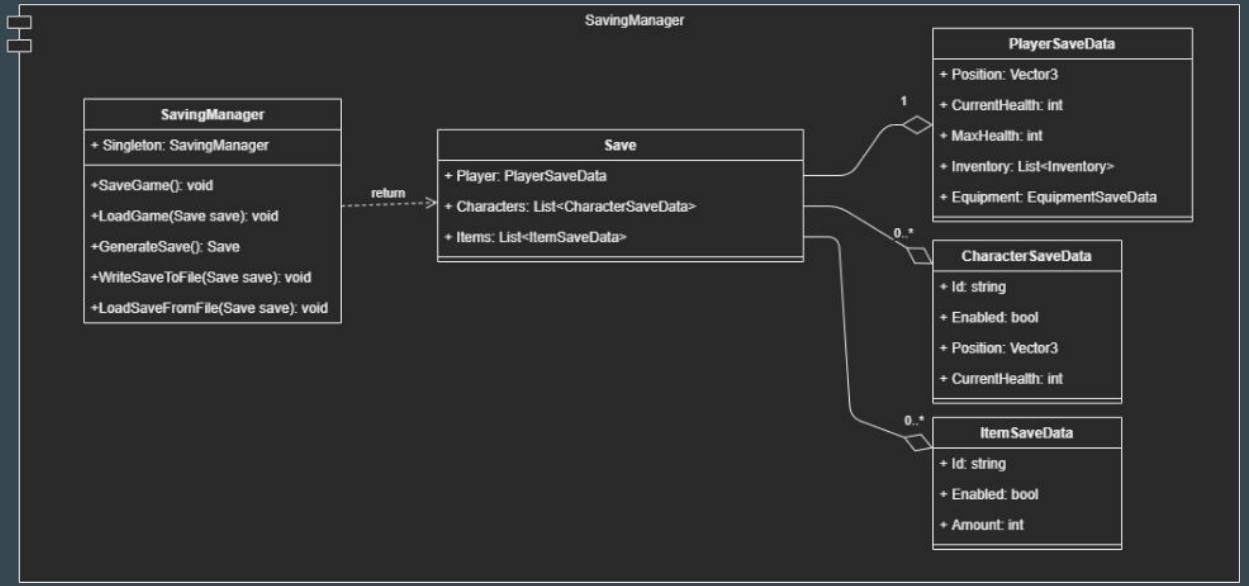
Modules:

- *Characters*
- *Items*
- *State Machine (AI)*
- *Interactive Items*
- *User Interface*



SavingManager

- Create "Save" object
- Convert "Save" to json format
- Write json to file



Klasa Save

```cs
using System.Collections.Generic;
using UnityEngine;

namespace Source.Core.SavingManager
{
    [System.Serializable]
    12 usages  Jaroslaw Kaszczak  1 exposing API
    public class Save
    {
        public PlayerSaveData player;  Serializable
        public List<CharactersSaveData> characters;  Serializable
        public List<ItemsSaveData> items;  Serializable
        public Vector3 cameraPosition;  Serializable
    }
}
```

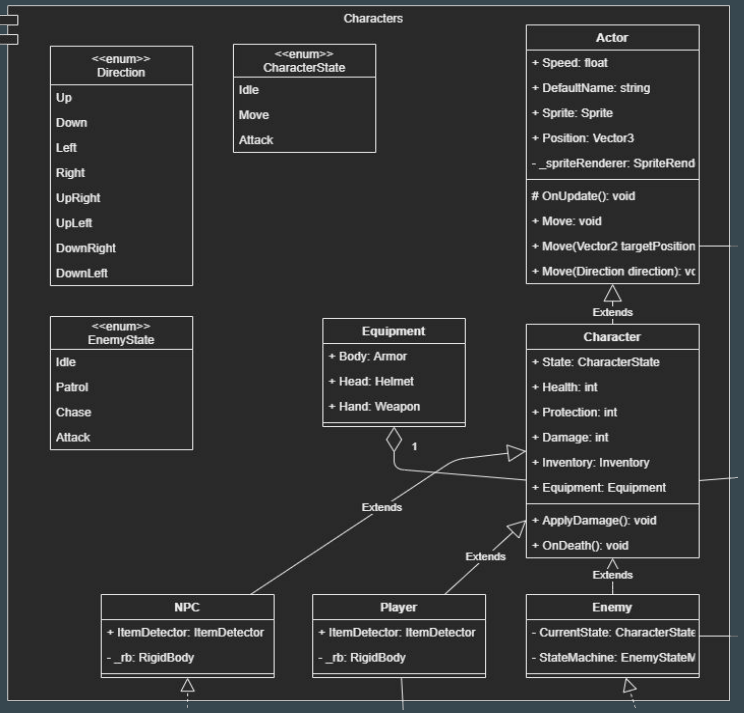
PlayerSaveData

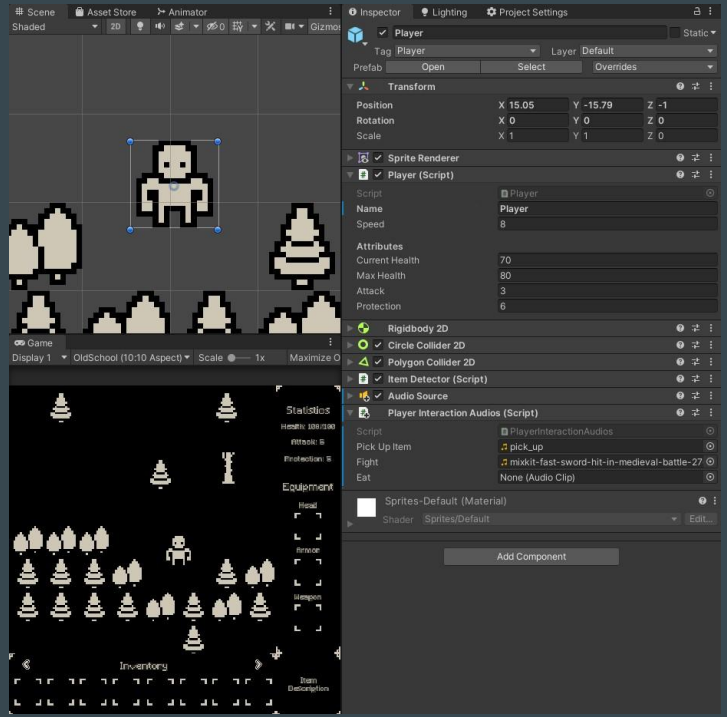
```cs
using System.Collections.Generic;
using DungeonCrawl.Actors.Items;
using Source.Actors.Characters;
using UnityEngine;


namespace Source.Core.SavingManager
{
    [System.Serializable]
    9 usages  Jaroslaw Kaszczak +1  2 exposing APIs
    public class PlayerSaveData
    {
        public Vector3 position;  Serializable
        public int currentHealth;  Serializable
        public int maxHealth;  Serializable

        /// <summary>
        /// Id's of the items
        /// </summary>
        public List<string> inventory;  Serializable

        /// <summary>
        /// Id's of the items
        /// </summary>
        public EquipmentSaveData equipment;  Serializable

```

json

```json
{
  "player": {
    "position": {
      "x": 15.050000190734864,
      "y": -15.789999961853028,
      "z": -1.0
    },
    "currentHealth": 80,
    "maxHealth": 80,
    "inventory": [],
    "equipment": {
      "helmet": "",
      "armor": "",
      "weapon": ""
    }
  },
  "characters": [
    {
      "id": "e9f948cf-98c6-47ca-9b64-5a45fb309efd",
      "enabled": true,
      "position": {
        "x": -9.25,
        "y": -4.420000076293945,
        "z": -1.0
      },
      "currentHealth": 50
    },
    {
      "id": "e0aae7f9-40be-430d-ab3a-594774ef940a",
      "enabled": true,
      "position": {
        "x": 1.8600000143051148,
        "y": -2.6500000953674318,
        "z": -1.0
      },
      "currentHealth": 20
```

Inheritance vs Composition

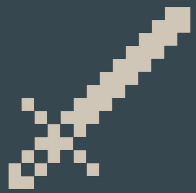




Using properties to update UI


```
public class Equipment
{
    /// <summary>
    /// Character witch have equipment instance.
    /// </summary>
    private Character _character;

    ? usages  ? overrides  Jarosław Kaszczak  ? ext methods  ? exposing APIs
    public Equipment(Character character)
    {
        _character = character;
    }

    private Armor _armor;

    ? usages  ? overrides  Jarosław Kaszczak  ? ext methods  ? exposing APIs
    public Armor Armor
    {
        get => _armor;
        set
        {
            if (value != null)
            {
                if (_armor == null)
                {
                    _character.Protection += value.Protection;
                }
                else
                {
                    _character.Protection -= _armor.Protection;
                    _character.Protection += value.Protection;
                }
            }
            else
            {
                _character.Protection -= _armor.Protection;
            }

            _armor = value;
            InventoryManager.Singleton.DisplayEquipment();
        }
    }
```

```
List<Challenges>
{
    “Unity”,
    “WiFi”,
    “Lockdown”,
    “PC Storage”,
    “Files exchange”,
    “Json”,

};
```



Fu<T>ure Plans

- Code refactoring
- Storyline
- Creating more advance futures in the Game
- Working on improvement of the Game
- Publishing Cat’s Minion on the Steam
- Writing tests for majority of the code

```
if (Question != null)
   Answer();
else
   “Thank you for watching”;
```