**Game documentation**

Contents

1. [Introduction](#Закладка1)
2. [Specification](#Закладка3)
3. [Gameplay](#Закладка4)
4. [Front-end](#Закладка5)
5. [Code description](#Закладка6)

Introduction

Next, you can read information about a game called "Mercenary". This document is intended as an orientation to the game itself and a more detailed understanding of it.

Specification

The aim of the game is to sneak into the castle treasury. The protagonist is a mercenary from a poor town, whose lord has been collecting huge taxes for a long time, thus ruining the lives of ordinary people. The protagonist decides to teach the arrogant lord a lesson and fill his own pockets.

Gameplay

The player can quickly move with the shift key, move normally (walk and jump) and fight with a sword. Loot drops randomly from enemies, as well as from some boxes. At each level, the player's task is to find the key and the exit, use the key to the exit and move on to the next location. The player has an inventory with cells, each of which has two buttons: green - treatment, orange - throw from inventory. There is also a task window, where the player's current goal is written.

Code Description

Enemy code:

EnemyController - the main code of any enemy, which contains information about it and basic functions (for example, getting hurt)

EnemyDamage - code for enemies that deal damage on impact.

EnemyPathfinding - is responsible for enemy movement.

MeleeEnemy - is responsible for enemies that deal damage in a certain area, relative to themselves, if the player steps into it.

Patroling - is responsible for enemy movement from point to point. (often used with pursuit).

Wandering - is responsible for the chaotic movement of the enemy (often used with pursuit).

Chasing - when the player is within a certain radius of the enemy, the enemy begins to pursue him until the player escapes from this area.

Inventory code:

Spawn - responsible for spawning items from the inventory (but not from enemies)

PickUp - responsible for adding items to the inventory.

Inventory - stores references to cells and their contents, also responsible for hiding and opening the inventory.

Cell - responsible for each individual cell in the inventory and for interacting with them.

Objects:

LootSpawn - responsible for loot spawning.

Looting - class for objects that store loot (for example, boxes).

Loot - structure that stores data about a specific loot.

Interactable - abstract class used for objects that can interact. (if they can be interacted with in the game, an icon will appear showing which button to interact with).

Healing - for objects with the healing property.

Parallax is used to create a parallax effect with a background.

Player:

Sword - responsible for combat with a sword.

PlayerHealth - responsible for the player's health (damage, heal).

PlayerData - stores player data (for saving and loading).

PlayerController - the main player code. Responsible for movement, saving and loading player data, displaying player data and his death.

KnockBack - responsible for knocking back enemies when they are hit.

Quests:

QuestCloud - responsible for handing out tasks that are displayed when approaching a certain object.

Quest - responsible for accepting the items requested by the task, changing tasks if there are several of them, and for a reward (if any).  
  
  
UI

VolumeManager - responsible for playing music and sounds and adjusting their volume.

SceneChange - responsible for moving from scene to scene.

MissionSButton - opens and hides the task window.

HealthBar - responsible for the large player health window (more precisely, updates the data about his health and displays it).

GameMenu - responsible for the functionality of the buttons in the game, mainly for the visibility of objects (menu, inventory and task window).

Extention - contains additional functions for code readability in other classes.

SavingSystem - responsible for saving data about the player.

Orekhovych Anastasiia