Game documentation

Contents

Introduction

Next, you can read information about a game called "Last straw". 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.

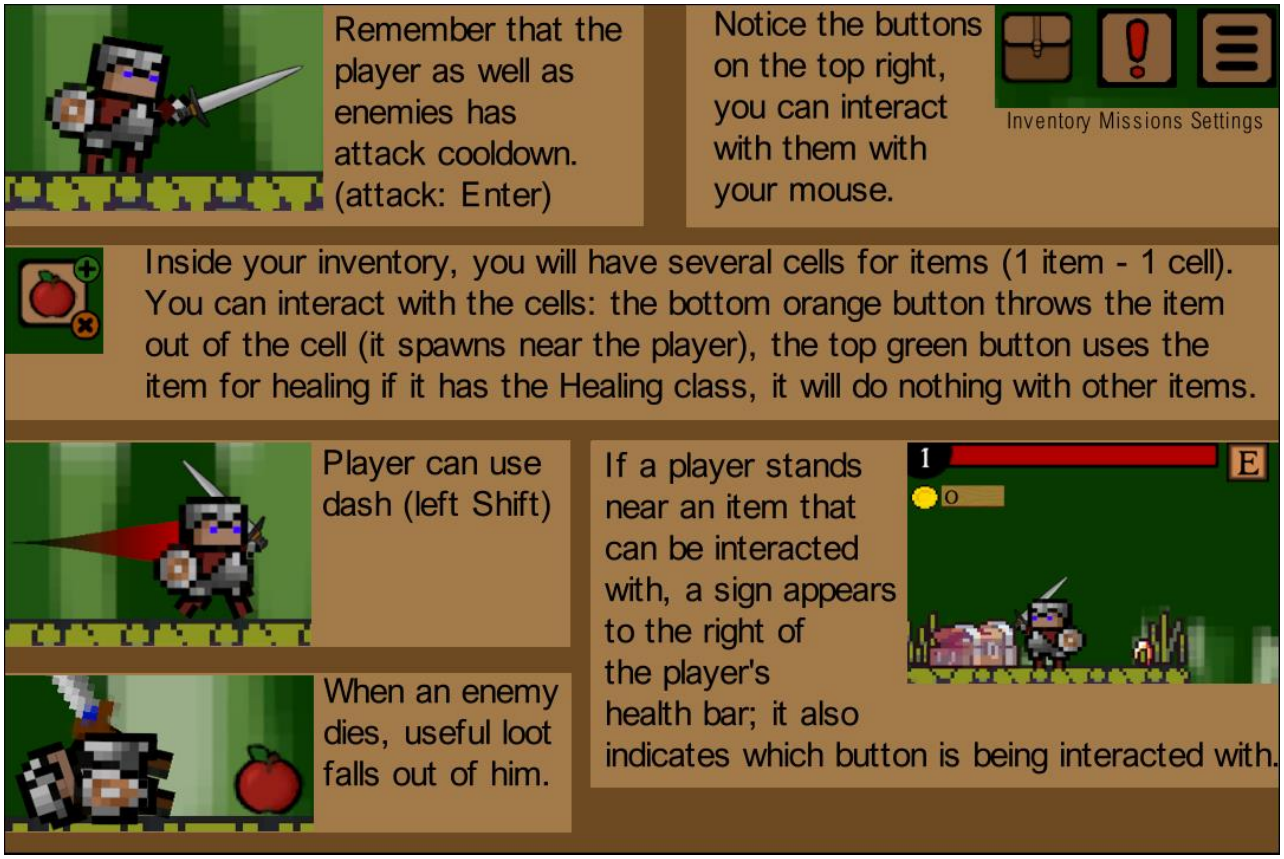
Gameplay

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. Spawning of items is random, so sometimes items may not drop at all, everything is fine, it is necessary. 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. To complete the task (the door will ask for a key), you need to find the key on the map, it will drop as loot, and throw it out of the inventory near the door. 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.

Game controls

The game starts from the Menu scene that is, the game starts with the launch of the Menu scene. During the game, no parameters need to be changed, and the controls cannot be changed either. During the game, files with player data (health, position, etc.) will be saved on the computer (Application.persistentDataPath + "/savings" in binary format). Game controls: buttons are used exclusively with the mouse, starting from the Forest scene, it will be possible to control the character:

walking - "A, D" and "left and right arrows", jumping - "space", jumping - "left shift", attack - "Enter", the buttons at the top left and in the inventory (buttons on the slots) will also be clickable. To open the inventory, you need to click on the bag icon, the inventory with slots will open, each of which has 2 buttons, the upper green button uses the item if it has the Healing property, if the item does not have this property, then nothing will happen, the lower orange one throws the item out of the inventory and spawns it near the player, pressing the button again closes the inventory. The button with an exclamation mark opens the panel with tasks of the current location, pressing the button again closes it. The rightmost button, with three stripes, opens the game menu, where there are sliders for controlling the volume of the music, you can also exit to the main menu from there, there are also save and load buttons, but they do not work as intended yet. To return to the game from this menu, you need to press the "Return to game" button.



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.

Tests

The program was tested:

- for smooth collision with platforms (tested during the game)
- for spawning items in a place accessible to the player (tested during the game)
- for convenient sound adjustment (tested during the game)
- for correct display of health (tested during the game)
- for correct interaction with buttons (the keyboard no longer affects them) (tested during the game)
- for correct completion of the task (now you can put the key in the door until the barrier falls off the door, but do not use it until the barrier disappears) (tested during the game)
- for turning enemies in the direction of their movement (tested during the game)

There is a known problem with enemies: if an enemy, while pursuing the player, falls from the platform where his patrol points were, then in an attempt to get to them, he will twitch from side to side under the point he needs.

Installations

The program has been made in Unity 2022.3.14f1 (Windows, Mac, Linux), and some packages needed to be installed for its correct work.
Needed packages (install in "window -> package manager"):

1. Cinemachine
2. 2D Tilemap Editor
3. 2D Tilemap Extras
4. Burst
5. Collections
6. Mathematics
7. Test Framework
8. Timeline
9. Unity UI
10. Version Control
11. Visual Studio Editor
12. 2D
13. Custom NUnit

Author: Orekhovych Anastasiia