Programmer assignment: Inventory system - part 2

Instructions

This task comes with a lot of optional ("Bonus") features. They are marked with the starting “Bonus” text after the feature number. If a feature does not have the “Bonus” text after the number, even between “Bonus” features, it is an obligatory feature. Each additional Bonus feature will be appreciated and valued.

Syntax instructions:

- Fields and methods
  - Public – use CamelCase notation
  - Protected and private – use _variableName notation
  - Local variables – use variableName notation
- Enums – use enum EnumNameType{ EnumValueA, EnumValueB} notation
- Constants – use CONSTANT_NOTATION

Unity instructions:

- Use Unity version 2019.2.17f1

Notes

- there is no need to waste time drawing your own inventory items/character
- you can use any graphics you find useful from online or any other sources
- The executable file will be tested on various aspect ratios ranging from 4:3 to 21:9, and must work (UI must scale) on all horizontally oriented aspect ratios of the window
- Advice: check the accessibility, functionality, and testability of all the data you send from another computer (builds, input touch, means to access analytics and repository)

IMPORTANT!
In your response mail you should provide the link to a google drive **folder named “Exordium_ProgrammerAssignment_dd.mm.yyyy”** with the following content:

- **“Exordium_ProgrammerAssignment_Level1_InventorySystem_dd.mm.yyyy.docx”** - a copy of this document, named**”**, with a green highlight of the text for each point you solved completely, yellow for partially, and red for unsolved.
- **InventorySystem.zip** containing a .zip of the "Assets" and "ProjectSettings" Unity project folders
- **Folder “Builds”** containing the following:
  - **InventorySystem_64.zip** containing the windows 64-bit standalone exe build and data folders

    - **InventorySystem.apk** Android build
    - **InventorySystem_WebGL.zip** containing WebGL build
- **InventorySystem_README.txt** which explains the testing and grading guidelines for your solution - buttons and input mappings, and which script is relevant for each subtask point.
- All guidelines from the task “Subversioning Development” of the previous assignment guidelines still apply

Test cases from the previous assignment apply, extended with the added capabilities of this assignment.

For all questions and unclarities about the assignment, contact andrija.stepic@gmail.com

1. Update Player Camera

✓ 1.1. The camera must follow the player with a delay in 2D space

✓ 1.2. Add the option to interpolate the camera position to an object, camera stays there for a few seconds,and then it moves back to follow the player

✓ 1.3. Prevent the player’s input while the camera moves

✓ 1.4. Add at least 1 triggerable object in the scene to test this feature.

2. Player character can pick up items from the ground to the inventory

✓ Add at least 2 of the following options to your previous implementation

✓ 2.1.1. Option 1: Pick up items by physics trigger collision - on input

✓ 2.1.2. Option 2: Pick up items by player position + physics overlap circle - on input

2.1.3. Option 3: Pick up items by physics circle casting in movement direction - on input

✓ 2.2 Add the possibility to switch between all the implemented options during runtime.

3. Update UI

✓ 3.1. Adjust ui from the previous and this assignment to support aspect ratios from 4:3 to 21:9

4. Update Equip Screen

✓ 4.1. Add a 5th slot for items - boots

✓ e4.2. Add a 6th slot type, but 6th and 7th slot in the equip screen - left and right ring

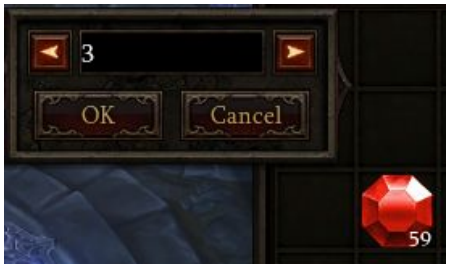
✓ 4.3. Add at least 2 examples of different boots items to pickup and equip

✓ 4.4. Add at least 2 examples of different ring items to pickup and equip

5. Update Item Interaction

✓ 5.1. Stackable items can be split into 2 stacks of varying amounts by the split screen that appears upon interacting with the grid cell containing the stackable item. The window contains the left and right button, editable text field and buttons "ok" and "cancel"

✓ 5.1.1. Split window contains a slider.



1. Stackable item split screen example image – Diablo 3

✗ 5.2. Bonus: mouse hovering + key shortcuts for splitting, dropping, equipping and unequipping an item

6. Update Items

✓ 6.1. Add durability to each equippable item - current and maximum

✓ 6.2. Current durability is spent only on equipped items per x units of the Player's walking distance.

✓ 6.3. When the durability of an item expires, the item is destroyed permanently.

✓ 6.4. Add the option to inspect the item's durability in the tooltip window

7. Update Attributes and Attribute UI window

✓ 7.1. Add a 5th type of attribute to the player - Luck. All definitions from the previous assignment part for an attribute still apply

✓ 7.2. Add Luck attribute change to at least 1 item which can be picked up

✓ 7.3. Display Luck attribute value in the Attribute UI window

8. Spendable attributes and Buff system

✓ 8.1. Add 2 more attributes to the player, health and mana, which are considered spendable attributes, unlike attributes listed in assignment 9.1., and as such, health and mana have maximum and current value, where the current value can be spent.

✓ 8.2. Items, when equipped, change the maximum or current value of attributes, by amount or percentage

✓ 8.3. Consumable items, when used, change attribute values in the following manner:

✓ 8.3.1. Hold bonus value over time (example: +30% strength over 4 seconds)

✓ 8.3.2. Ramp value up and/or down over time (example: change dexterity from 0 to +25 in 2 seconds, and hold +25 for 5 seconds)

✓ 8.3.3. Change value over time in tick manner (example: Damage Health by 20 over 3 seconds / Heal health for 3 per second over 6 seconds)

✓ 8.4. Create 1 Consumable item for each example change type, include them in spawning so they can be applied on the Player

✓ 8.5. Bonus: Display active buff duration in UI as Filled Radial360 image, support multiple active buffs display simultaneously.

9. Mobile touch input

Implement your own or add an external plugin for mobile touch input of your own choice, and implement the following options in the same unity project

Gestures - reactions

✓ 9.1. Long press - moves the item “into the air”

✓ 9.1.2 - on long press stop (finger release), if item was above empty inventory slot, it is slotted into that slot, if item was above the world space (there is no UI under the release position), item is dropped into the world, otherwise item is returned to the slot from which it was taken "into the air"

✓ 9.2. Touch up - releases the item from “the air”

✓ 9.3. Double tap - on an item equips/unequips equipable item / uses usable, consumable item

✓ 9.4. Pinch gesture - zooms the camera in and out between zoom limit values

✗ 9.5. Bonus: Pan gesture - if started over an item that can be split, and if ended over an empty slot or slot containing the same type of item - opens the splitting screen

✓ 9.6. Tap gesture - on the item in the inventory - shows the item tooltip

✓ 9.7. Implement one of the player character pick-up options so they can be picked up in the mobile build of the project, if necessary and if you choose so, you may add another UI button for “pick-up” interaction

9.8. Movement - choose one of the following:

✓ 9.8.1. Tap gesture - in the world - moves the character to the pressed world space location - choose one of the following

✓ 9.8.1.1. Movement is linear, from the current player position to the destination, and stops at colliders

9.8.1.2. Movement uses path-finding to reach the destination

✓ *9.8.2. Implement a mobile thumbstick/joystick input which will serve as horizontal/vertical input*

✓ 9.9. Bonus: if you chose 9.8.1., prevent movement by tap, if tap happens over the UI

10. Analytics

✓ 10.1. Implement in-game analytics with a plugin of your choice (Unity Analytics is a valid solution if you consider it's suitable for all the requirements) for PC and mobile build.

✗ 10.2. Bonus: Implement in-game analytics for WebGL build.

✓ 10.3. Implement analytics events for the following:

✓ 10.3.1. On each build startup (platform, local time)

✓ 10.3.2. On Item Picked up (item name, item type)

✓ 10.3.3. On Item Equipped (item name, item slot)

✓ 10.3.4. On item Used/Consumed (item name, item type)

✓ 10.3.5. On window opened (window type)

✓ 10.3.6. On Player moved each 10 unity units total, with a maximum of 5 events for a single build execution.

✓ 10.3.7. On Player collided with the in-scene collider, with a timeout of 5 seconds minimum between 2 analytics sending events.