Starting up:

Whenever I start a project I look at the features that are required. In this project I was tasked with these:

• Talking with NPC

• Interacting with the world

• Buying and selling items

• Displaying item icon and price

• Equipping items

• Update Unity to version

I started with updating Unity. Then just like with most projects, I grabbed a piece of paper, and began designing the game. Deciding which type of animation to use was quite a challenge and it took a bit more time than I planned. The version I came up with allows you to create an animation once, while also giving player an option to freely replace parts of character's body.

Coding:

I started with writing down the list of necessary scripts:

• **Player Movement** – used to move the player character

• **Skeletal Sprite Animation** – used to update character body parts

• **Dialogue Manager** – used to control dialogue menu

• **Inventory Manager** – used to control inventory menu

• **Shop Manager** – used to control shop menu

• **etc.**

Here is a more detailed description of the reasoning behind each class:

Dialogue

Dialogue component was something I implemented later along the line. It contains variables for:

- characterName(string),
- interactions(DialogueElement),
- jokeSetup(string),
- jokePunchline(string)
- and dialogueManager(DialogueManager)

The idea behind this component is to allow a GameObject to communicate with DialogueManager and display DialogueElements or string variables.

DialogueElement

Contains initial character line and two DialogueResponses for the player. DialogueResponse is a class that contains text(string), isPositive(bool) and interaction(UnityEvent).

DialogueManager

Used to control dialogueMenu, display messages and invoke interactions (UnityEvents).

InventoryManager

Used to control inventoryMenu and display inventory information.

MySpriteSheet

A type containing 4 Sprites from a Sprite Sheet(1 for each side) and BodyPart property (BodyPart – enum). Character skins are loaded based on this class and its BodyPart property.

NpcLogic

A component enabling NPC to use dialogue and shop.

Player

A component enabling player to interact with the world and be affected by it (inventory and skin).

Player Movement

I decided that the player should be able to move horizontally or vertically, but not diagonally. Usually I would create a specific component to change player controls, but here I thought that the classic Input.GetAxisRaw() based movement would suffice.

ShopManager

Component used to control shopMenu. It can display and update character inventory (it can be player's or NPC's).

SkeletalSpriteAnimator

Component used to control Sprites of each character part. It references:

- character current skin (CharacterSkinObject)
- each of character body parts containing a SpriteRenderer (Transform)
- spriteSheet for each of character's body part (MySpriteSheet)
- bodySprite for each of character's body part (SpriteRenderer), at Start() gets component from: each of character body parts containing a SpriteRenderer (Transform)

Every time player changes direction, currentDirection is updated (private variable, Direction enum). Then its value can be accessed via PlayerDirection property. For example, currentDirection value is used by Skeletal Sprite Animation component, so it can change Sprites properly.

Using Scriptable Objects

Introduction

Another important aspect was creating an Inventory system. I more or less had an idea of how it would operate, but to be honest, it was the first I ever implemented it in one of my projects. Anyway, to create this system, I decided to use scriptable objects. I created a template for:

• EquipmentObject (item you can equip)

• CurrencyObject (item used to buy equipment)

• InventoryObject (inventory which contains a list of equipment objects and 1 currency)

• CharacterSkinObject (contains MySpriteSheet for each of character body parts)

I had an idea of buying items with different kinds of currency, but decided to abandon it, since it was not neccessary. I also played a bit with SO based event architecture (GameEvents and GameEventListeners), but also used traditional references and update loops, so I ended up with a bit of an amalgamation.

Results

Here is a quick summary of the result of my work:

Talking with NPC

Player can engage in a dialogue with an NPC. Dialogue box waits 1 second before displaying each message. Dialogue can lead to a short conversation or trading.

Interacting with the world

Player can collide with the NPC or interact with it by clicking left mouse button.

Buying and selling items

Items can be bought from the NPC (shop appears as a dialogue interaction) and sold by opening your own shop (press TAB). Item icons are displayed.

Displaying item icon and price

Unfortunately, not yet implemented.

Equipping items

Items can be equipped by clicking the corresponding slot in the inventory menu. This will change the skin of the corresponding body part. If item is equipped, clicking again, will unequip it and change the skin of the corresponding body part back to default. Also happens when equipped item is sold.

My opinion

As far as prototypes go, I think I did quite well. I think most of the code I wrote is solid, but I could definitely improve visual aspect of the project. If there are any bugs, they are mostly visual (wrong animation hook-ups, etc.). To be honest, I wanted to do so much more, but I already abused additional time. Despite that, it's been fun developing this project, and I'm looking forward to your reply.