Name: Kim Huynh Mark \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/50

## Brief introduction \_\_/3

I am working on several different features. The first feature I was working on was the player attack (although another team member helped complete it, making it not necessarily my own feature). The player attack consists of two “attack points,” or circle colliders, that deal damage to anything in its range when the player presses the spacebar to attack. The main basics of the functionality are already completed.

The second (and current) feature I oversee is the loot and vendor system. I create a shopkeeper NPC that is just a sprite surrounded by a big box collider. Once the player comes in range of the box collider, a shop menu will pop up with different items to purchase. In the future, items will be pulled randomly from a database to supply the shopkeeper menu. An example of some items the NPC might sell will include health potions, weapon upgrades, and potions that enhance the player’s stats (speed, weapon speed, etc). Potions can be consumed later and is stored in an inventory-like space. Weapon upgrades apply to the character immediately. As long as the player has sufficient money to purchase an item (as detected in a Player Manager script), the appropriate amount of money will be deducted once an item button is pressed.

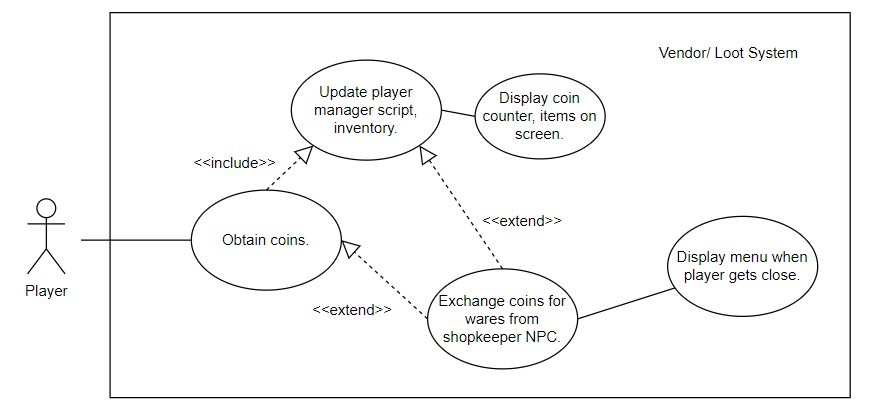
The third feature I would like to work on is the dialogue system, just to give a plotline to the game. I envision it to be like the shopkeeper NPC behavior. Once a certain area of the map is collided with (and is detected by a box collider), a dialogue box (prefab) appears with text to explain the story. This isn’t very important, however, and will not be the focus of this champion.

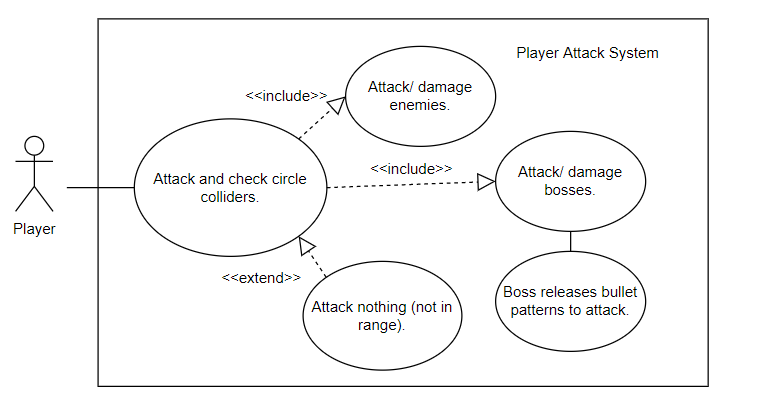
The fourth feature that I want to work on in the future would be coding the boss patterns for the bullet-hell portion of the fight. The details aren’t clear now, but I would have to make a bullet prefab and set some damage points to it. Then I could use physics to fling the bullets in different directions. That is but a wish of mine though. In this document, I will be focusing more on how it interacts with the player.

## Use case diagram with scenario \_\_14

In the SA presentation, a general grouping of features were bundled in a general use case. I’d like to explore more on the complex features of vending/ looting and attacking. The dialogue system is too simple to make a proper use case system out of it.

### Use Case Diagrams





### Scenarios

**Name:** Obtain coins.

**Summary:** The player obtains a coin, collects it, and may choose to use it for purchase.

**Actors:** Player.

**Preconditions:** Player Manager script is set up. Coin exists.

**Basic sequence:**

**Step 1:** Player collides with coin collider.

**Step 2:** Coin prefab sees if the colliding object has Player tag.

**Step 3:** If it does, increase total coins held in Player Manager script.

**Step 4:** Update UI coin counter on screen from referencing script.

**Step 5:** Deletes coin object.

**Exceptions:**

**Step 1:** The player encounters a shopkeeper NPC and steps into NPC box collider range.

**Step 2:** A menu prefab with available items (randomized from loot database) shows up.

**Step 3:** If the player can afford the item (based on current money amount) and presses an item button, the item will be purchased, and cost is deducted from money amount in player script (basically update Player Manager script).

**Step 4:** Display updated coin amount and any relevant purchased items on screen.

**Post conditions:** Total coins are updated.

**Priority:** 1\*

**ID:** C4\_5

\*The priorities are 1 = must have, 2 = essential, 3 = nice to have.

**Name:** Attack and check circle colliders.

**Summary:** The player potentially deals damage to a non-player enemy.

**Actors:** Player.

**Preconditions:** Enemy is initialized, exists.

**Basic sequence:**

**Step 1:** Player presses the spacebar on a keyboard.

**Step 2:** Attack animation starts and circle colliders detect if any objects with an enemy tag are in range.

**Step 3:** If enemies are in range of colliders that the player is facing, deal damage (set by Player Manager script).

**Step 4:** If enemy is a boss type, make boss release a delayed bullet in direction of player to retaliate (possible feature).

**Exceptions:**

**Step 1:** No one is in range. Don’t deal damage, but animation executes.

**Post conditions:** Enemy health may or may not decrease.

**Priority:** 1\*

**ID:** A5\_6

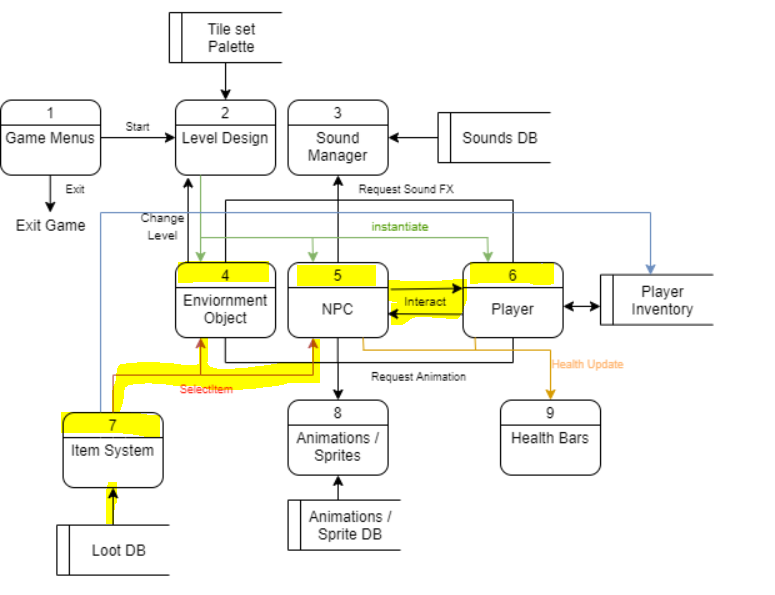
\*The priorities are 1 = must have, 2 = essential, 3 = nice to have.

## Data Flow diagram(s) from Level 0 to process description for your feature \_\_\_\_\_\_\_14

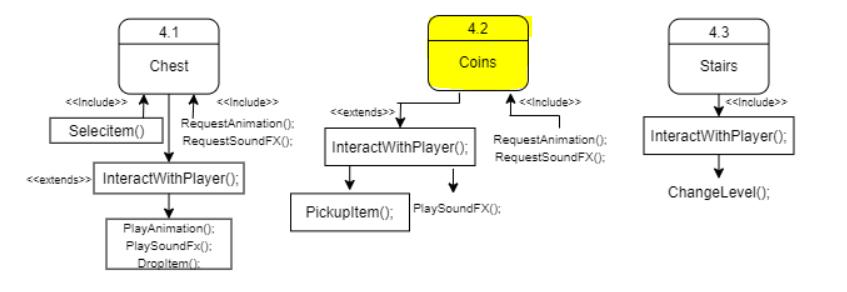
I oversee multiple features. Please see the data paths for 4.2, 5.1, 5.3, 6.2, 6.3, 6.5, 7.1.

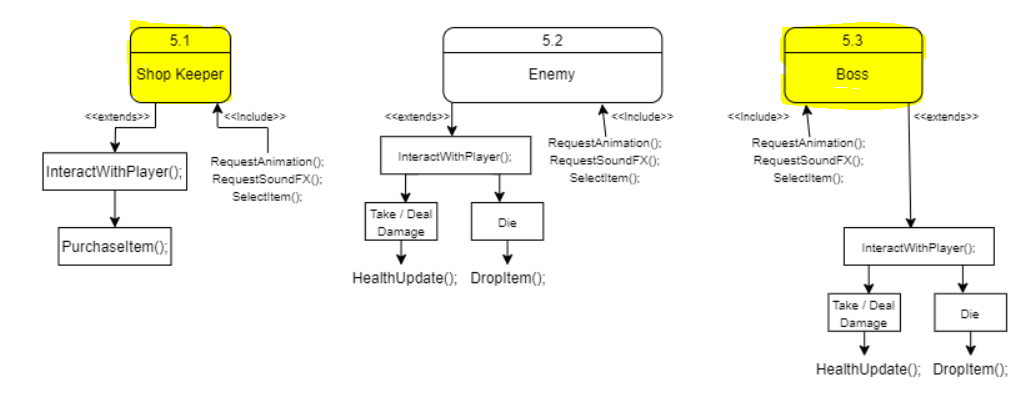
To save space, some unrelated data paths are omitted.

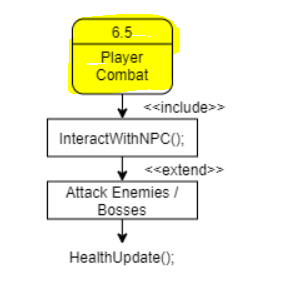
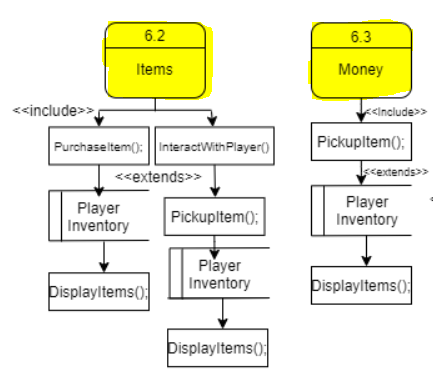
### Level 0

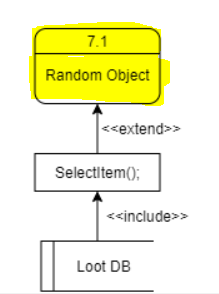


### Data Flow Diagrams









### Process Descriptions

**Coins (4.2):**

IF tag==player and enters collider:

Play sound effect, add coin to Player Manager script, and delete coin object from level.

**Shop Keeper (5.1):**

IF tag==player and enters collider:

Display menu prefab with randomized items taken from database.

IF item button is pressed:

Check coin amount in Player Manager.

IF coin amount>=item amount:

Deduct total coin amount with cost,

update amount (and any internal upgrades if things like weapon upgrade is bought) in Player Manager.

Store other items in inventory.

Potentially activate sound.

Display updated coin amount on screen.

**Boss (5.3):**

Play idle animation.

IF tag==player passes collider:

Play attack animation. Play sound.

Release bullet prefabs, distribute randomly with speed in 2D dimension. Player Manager can reference the bullet damage from this script.

IF take damage from tag==player:

Get player damage from Player Manager script.

health-=damage given by script.

IF health<=0:

Die animation start.

Delete boss object.

Start end dialogue function.

**Items (6.2):**

IF item is purchased from shopkeeper (with affirmation from shop keeper script):

Store in player inventory.

May display certain items (health potions) on screen.

Different potions may only show up in inventory menu.

IF tag==player and enters item collider:

Update storage script and add to inventory.

IF health potion:

Update total health potions held currently, display on screen.

**Money (6.3):**

IF tag==player and enters coin collider:

Increment total amount of money in Player Manager

(inventory).

Display updated coin amount on screen.

IF inventory is opened:

Display coin amount, items, etc.

**Player Combat (6.5):**

IF time elapsed >= next attack time:

IF spacebar is pressed:

Find direction player is currently facing (left or right).

Activate corresponding circle collider and find what is in range of it. For each enemy in collider, deal damage to them (reference their health with enemy script). If boss, reference boss script to deplete health.

**Random Object (7.1):**

IF sent signal from shopkeeper script:

Pick objects from item/ loot database based on random number generated.

Send picked items to shopkeeper script.

## Acceptance Tests \_\_\_\_\_\_\_\_9

**Vending/ loot system feature**

Have preset amount of money (10,000 coins). Buy different items 1000 times and output details (cost, item) to output file.

The output file should be like the following:

* Money spent on items is no greater than the money left in the inventory.
* The items bought \* their cost = (1000-leftover coins).

Once items and their prices are finalized, more concrete outputs can be given.

**Random item/ item database feature**

Trigger the random item feature 1000 times and send outputted items to output file

The output file should be like the following:

* A total of 1000 items.
* Items aren’t 100% identical and are valid items from item database.
* Prices match the item type (as displayed in the database).
* There’s enough variation in items to call it random.

Can also try triggering for three random items and check if it shows up properly in the shop keeper menu.

**Player attack feature**

Make the player attack 1000 times (with or without enemies, with hero facing left/ right). Record damage output, player direction into output file.

The output file should be like the following:

* The damage output showing up as the same number as set in the script.
* Directions that the player faced, and corresponding (and correct) circle collider chosen.
* Outputs if enemies were present and detected during the attack.

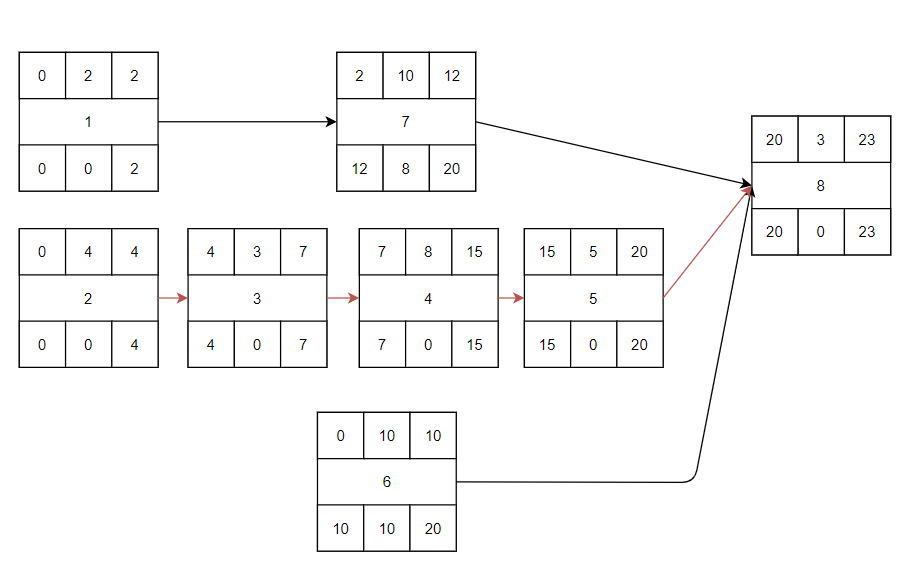
It is also possible to observe the behavior with Gizmos.

## Timeline \_\_\_\_\_\_\_\_\_/10

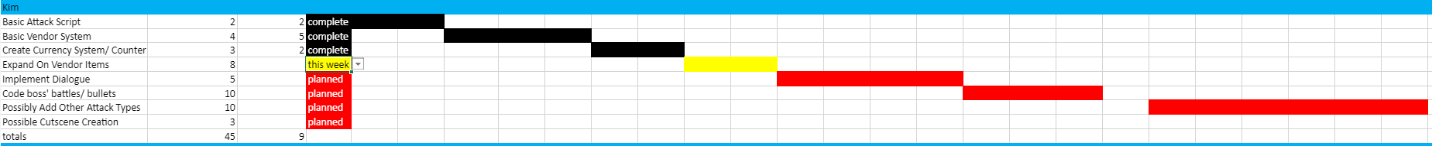
### Work items

|  |  |  |
| --- | --- | --- |
| Task | Duration (Hours) | Predecessor Task(s) |
| 1. Basic player attack script. | 2 | - |
| 2. Basic vendor system, menu. | 4 | - |
| 3. Currency system, $ counter. | 3 | 2 |
| 4. Expand vendor options, item usage. | 8 | 3 |
| 5. Implement dialogue. | 5 | 4 |
| 6. Code boss’ bullets. | 10 | - |
| 7. Possibly add other attack types for player. | 10 | 1 |
| 8. Possibly create endgame cutscenes. | 3 | 5, 6, 7 |

### Pert diagram



### Gantt timeline



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  | 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  | 3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 4 |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 7 |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 5,6,7 |  |  |  |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |