Spaghetti Systems Analysis

By Scott, Collin, Ben, Garrett, Jeffrey, Cole, Korben

The Story of the **Pastafarian's Delight**

In the distant realm of Al Dente, the annual Great Feast of the Flying Spaghetti Monster is approaching.

Every year, the great and benevolent Flying Spaghetti Monster descends from the heavens to indulge in a dish made by the most clever and worthy chef of the land.

This year, that chef… is you.

PASTAFARIAN'S DELIGHT STORY cont....

Your quest is to adventure into the pantry below, solving perplexing riddles and proving your culinary prowess. Each room of the pantry holds a clue to the fabled pasta dish.

But beware, the depths of the pantry contains unknown dangers.

Once all ingredients have been gathered, you must prepare the legendary dish. If you have chosen wisely, you may continue your journey to create more magical dishes.

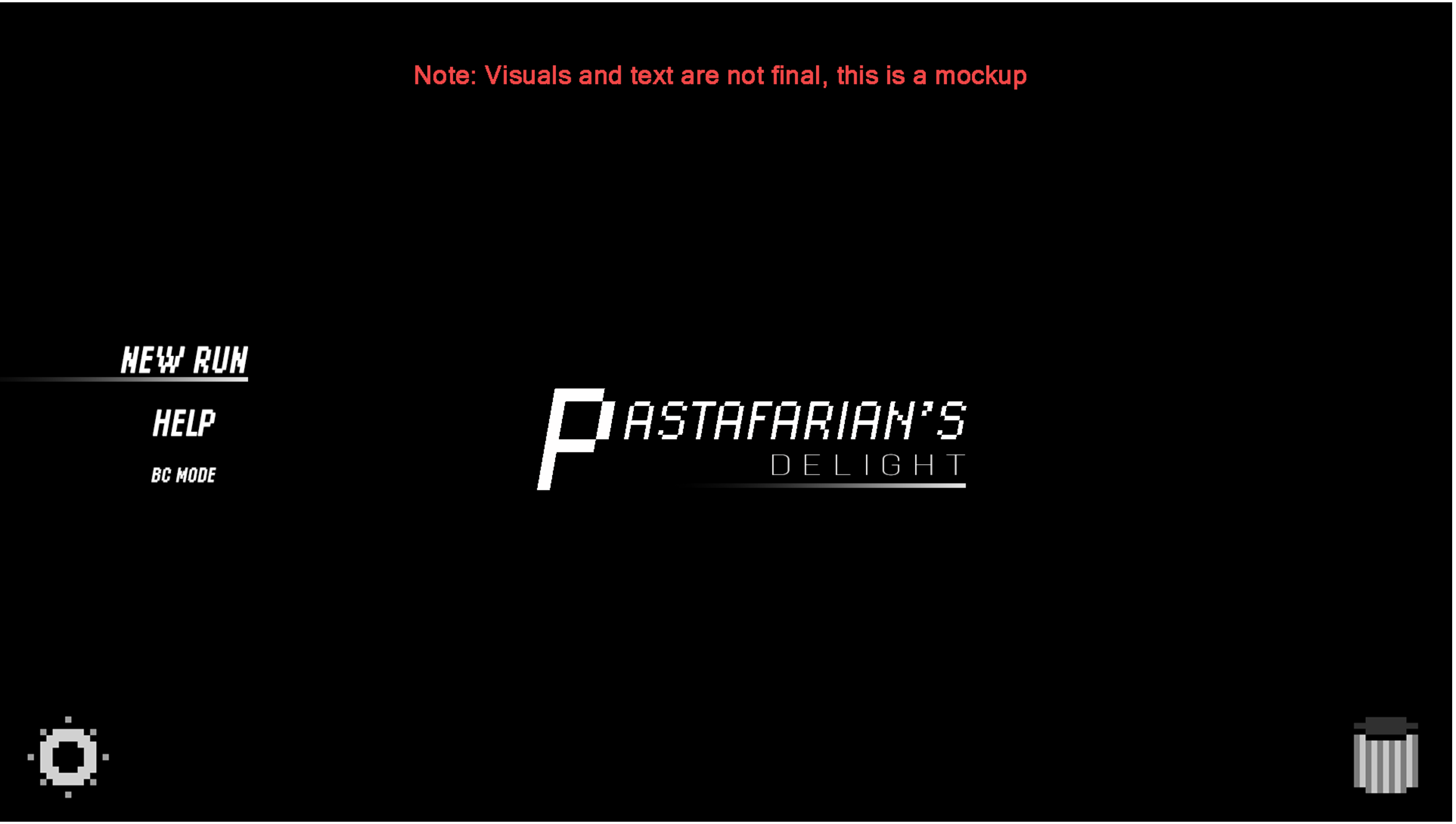
If not, you may never cook again…

Meet the team

Spaghetti Studio

- Garrett
- Scott
- Collin
- Korben
- Jeffrey
- Cole
- Ben

Scene 1: Launching the game, Main Menu

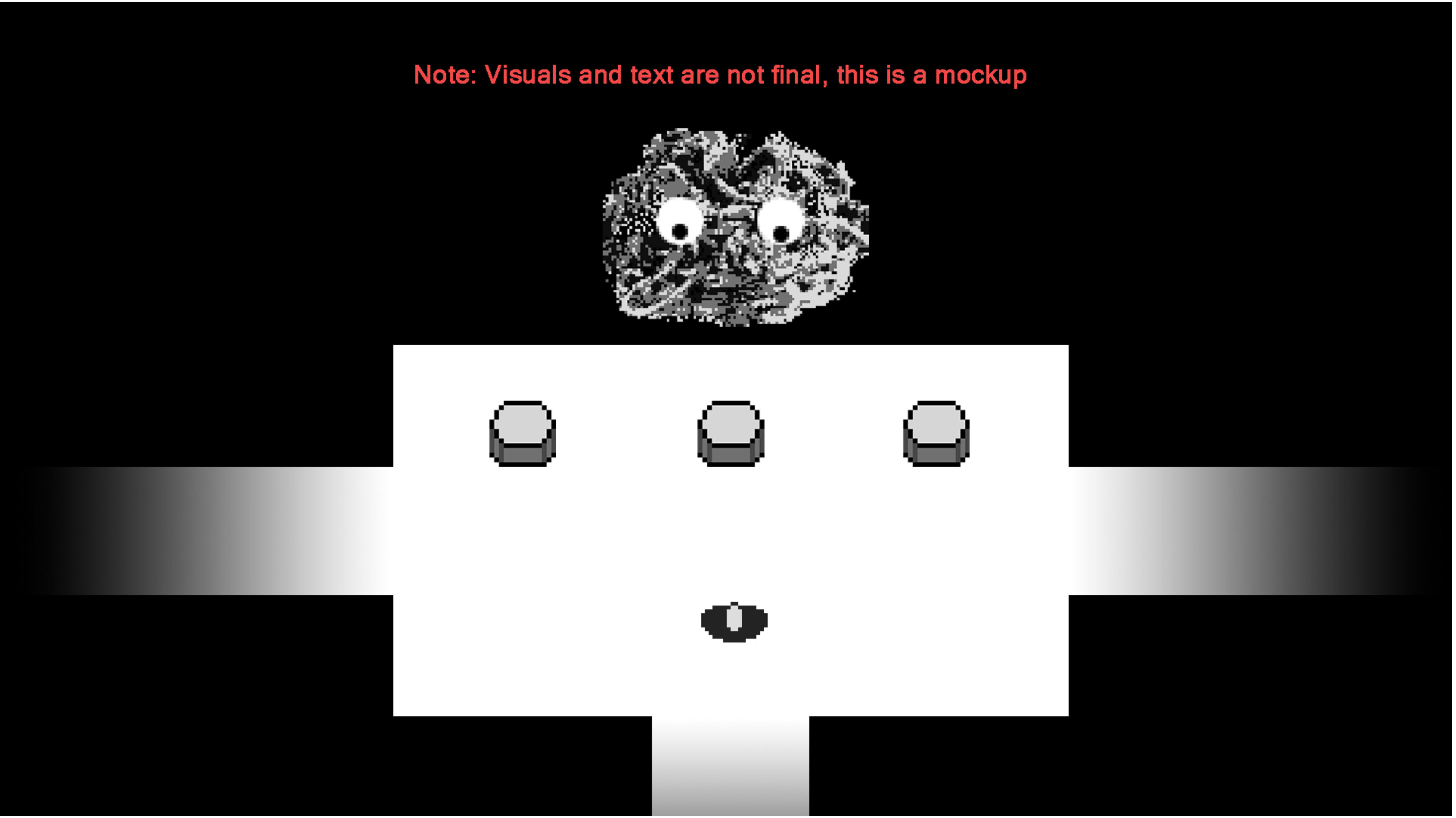


Dialogue:
None

Action:
Select New Run, Help, or BC Mode

Notes:
Soft background music
Sounds when hovering over or selecting menu items

Scene 2: Enter main room, meet the Flying Spaghetti Monster

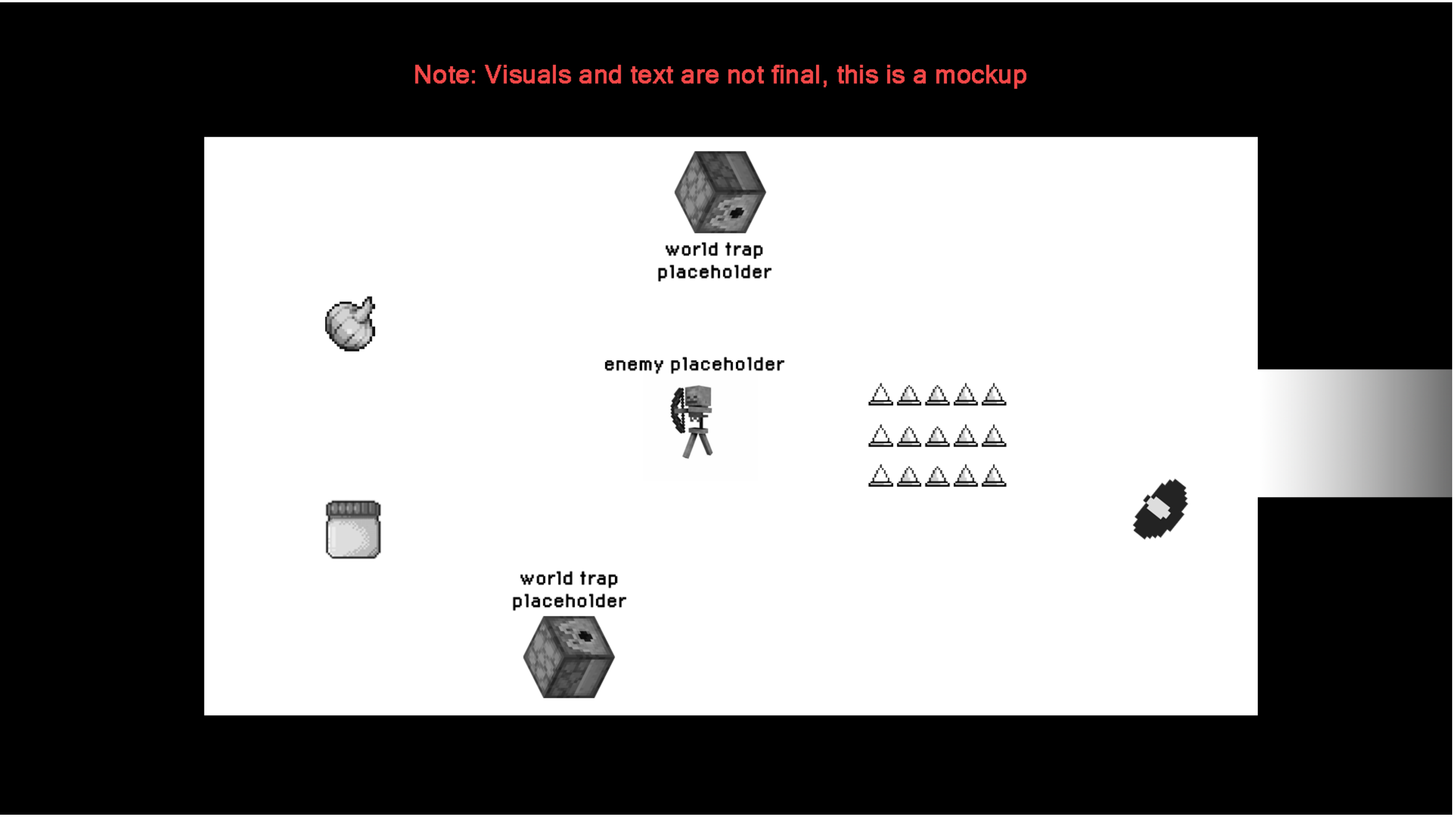


Dialogue:
"The Flying Spaghetti Monster has chosen you to cook him a pasta dish.
You must solve his riddles to learn the ingredients"

Action:
Player controls character
Player walks into one of the hallways

Notes:
Top-Down Camera
Soft background music

Scene 3: Enter a riddle room, defeat enemies and solve the riddle



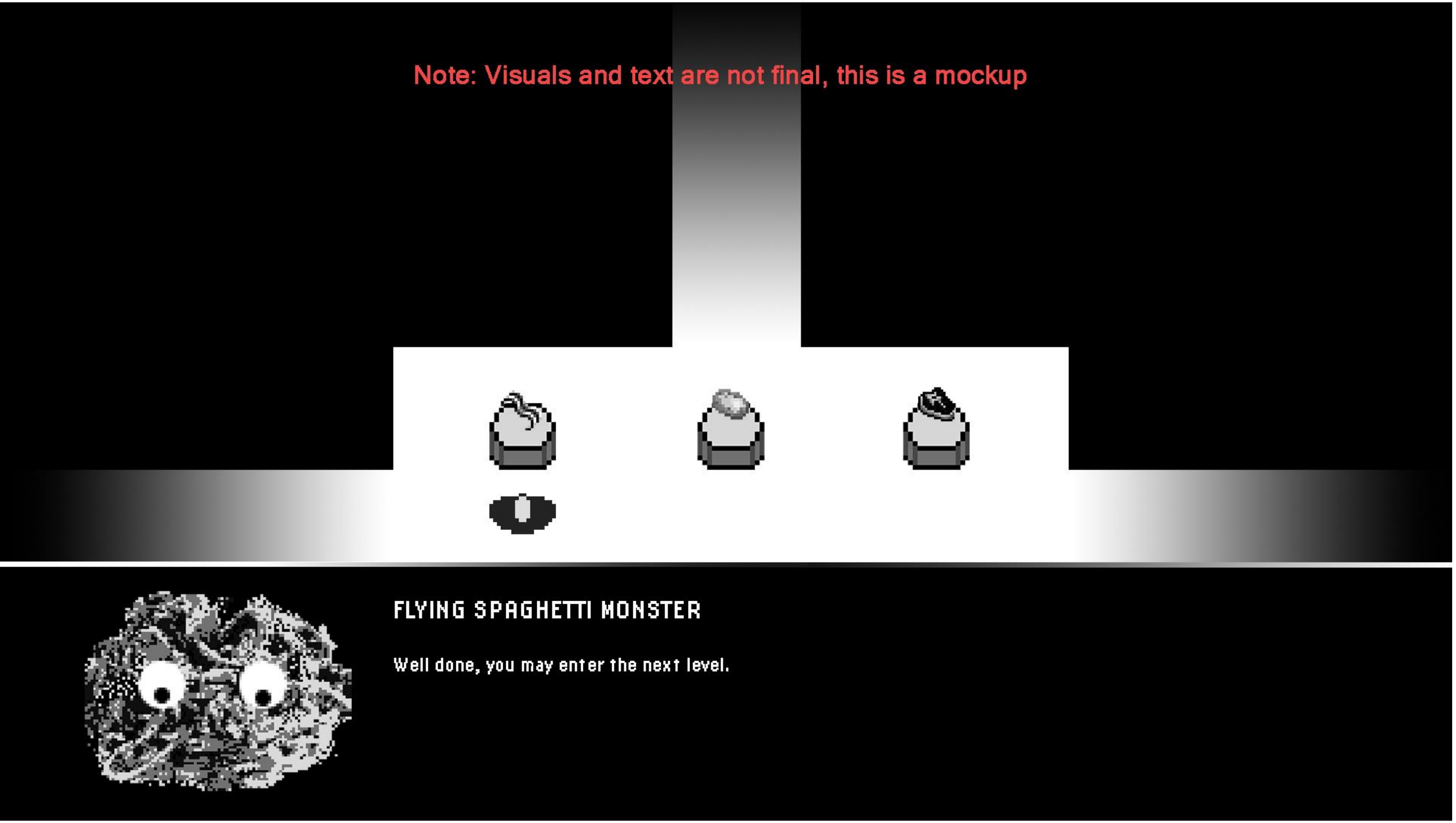
Note: Visuals and text are not final, this is a mockup

Dialogue:
None

Action:
Load monsters/traps
Player defeats enemies
Player guesses ingredient based on the riddle
Once room is completed, a powerup is dropped signifying the end of the battle
Player moves to next room

Notes:
Top-Down Camera
Action music
Level 1 contains 3 riddle rooms (3 ingredients)

Scene 4: Puzzle Solved, Go to next level



Note: Visuals and text are not final, this is a mockup

FLYING SPAGHETTI MONSTER

Well done, you may enter the next level.

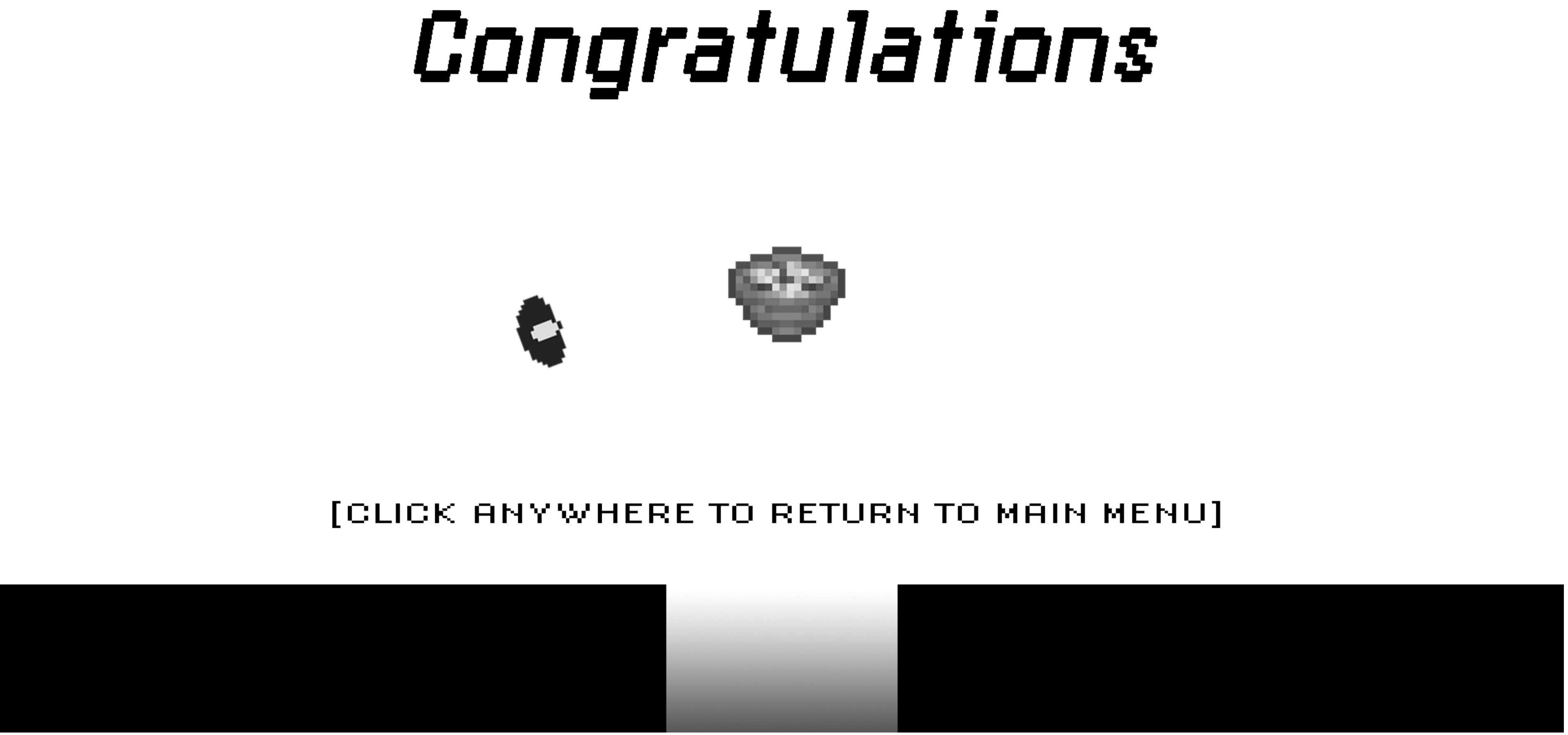
Dialogue:
Flying Spaghetti Monster tells you to submit your ingredients, if you are correct, then you can proceed to the next level

Action:
Submit the chosen ingredients.
If you are correct, you may proceed to level 2.
If you are incorrect, return to rooms to try and solve the riddles again.

Notes:
Top-Down Camera
Soft background music
Level 2 contains 5 riddle rooms.
Level 3 contains 7 riddle rooms

Scene 5: Solved all levels, Game Won

Note: Visuals and text are not final, this is a mockup



Congratulations

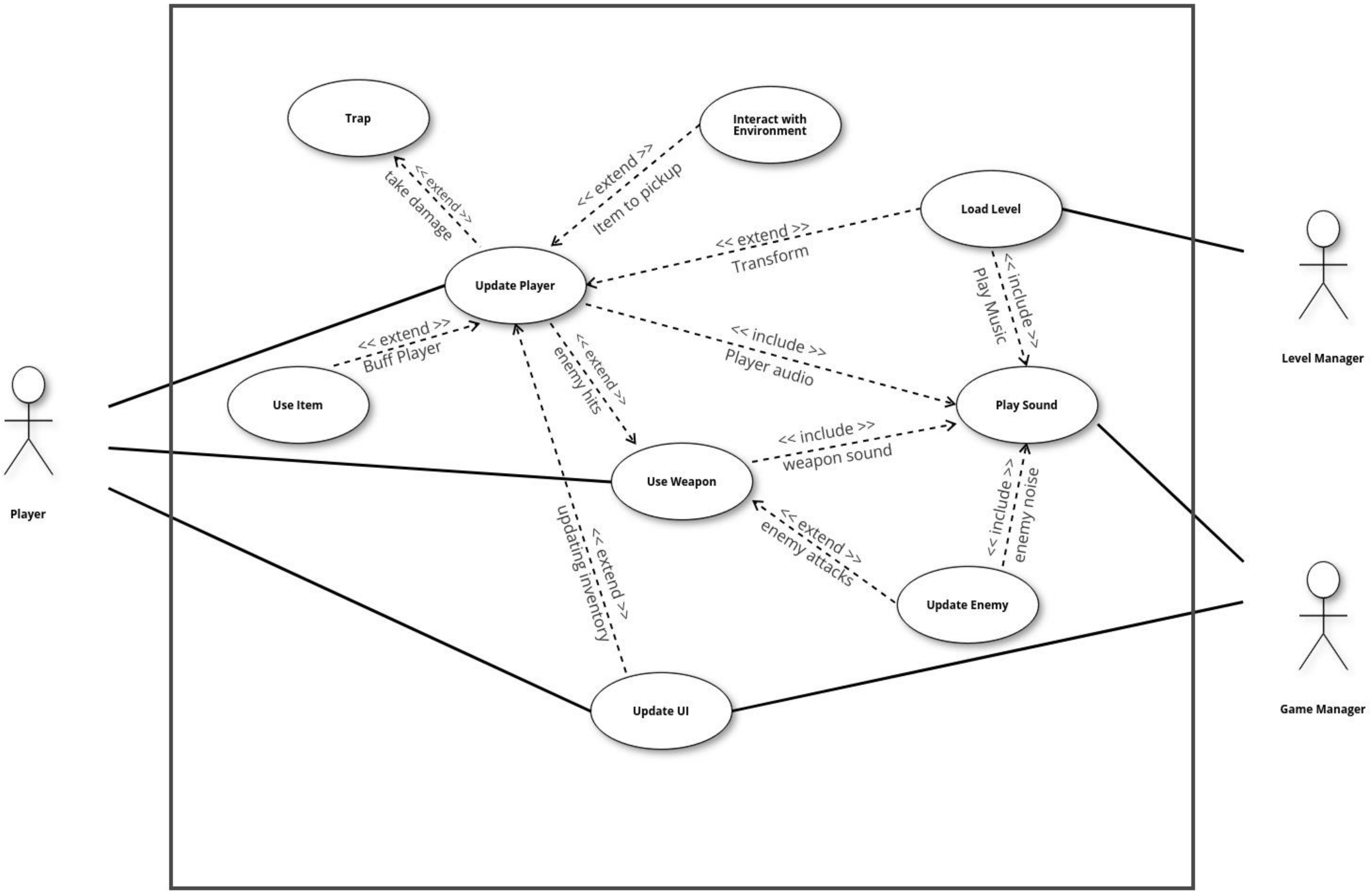
[CLICK ANYWHERE TO RETURN TO MAIN MENU]

Dialogue:
None

Action:
Player can click to return to main menu

Notes:
Victory music (copyrighted?)

Global Use Case



Context Diagram

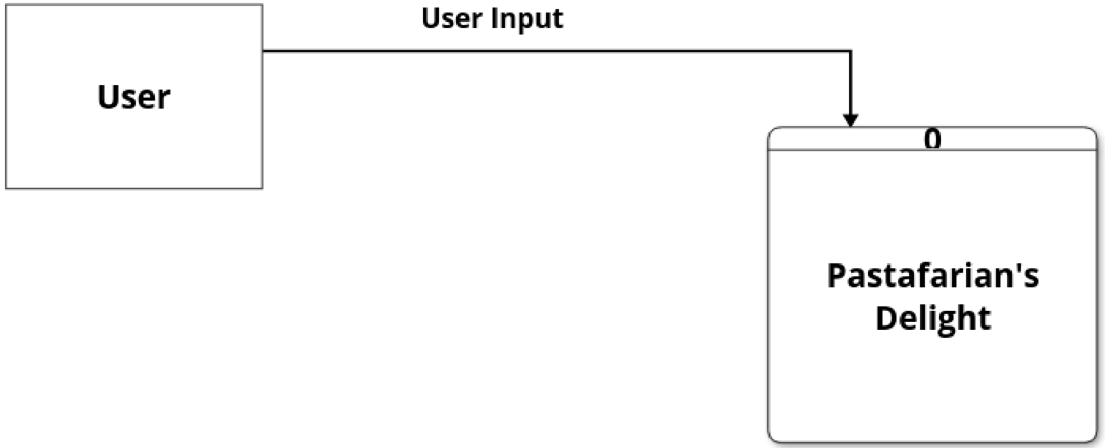
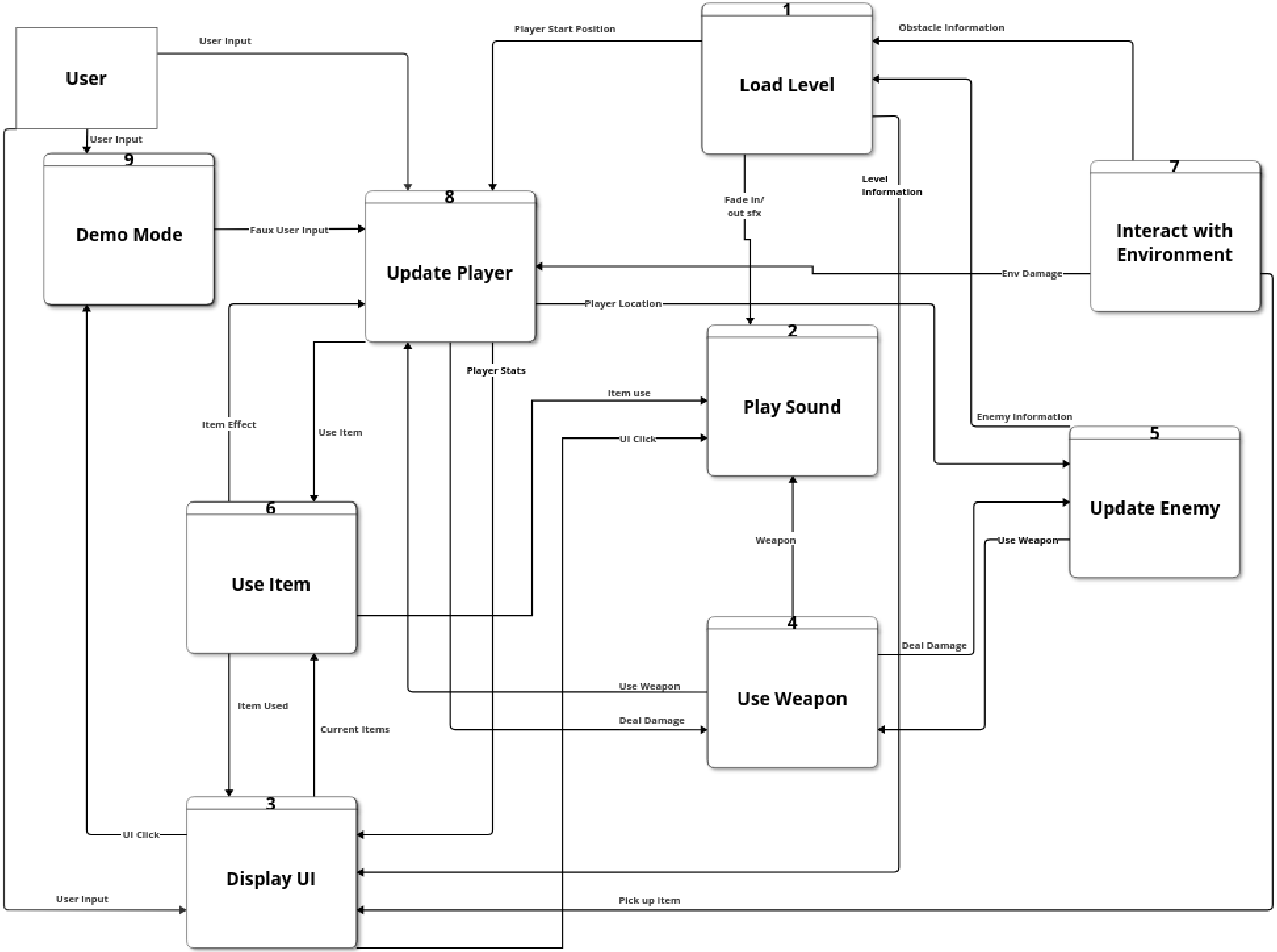


Diagram 0


User
User Input
User Input
1
Load Level
Player Start Position
Obstacle Information
Fade in/
out sfx
Level
Information
9
Demo Mode
Faux User Input
8
Update Player
7
Interact with
Environment
Env Damage
Player Location
2
Play Sound
Player Stats
Item use
UI Click
Enemy Information
5
Update Enemy
Item Effect
Use Item
6
Use Item
Weapon
Use Weapon
Deal Damage
4
Use Weapon
Use Weapon
Deal Damage
Item Used
Current Items
3
Display UI
UI Click
User Input
Pick up Item

Collin – UI / Inventory / Player

UI

- Load Game Menu
- Hud (health / level)
- Display Inventory

Inventory

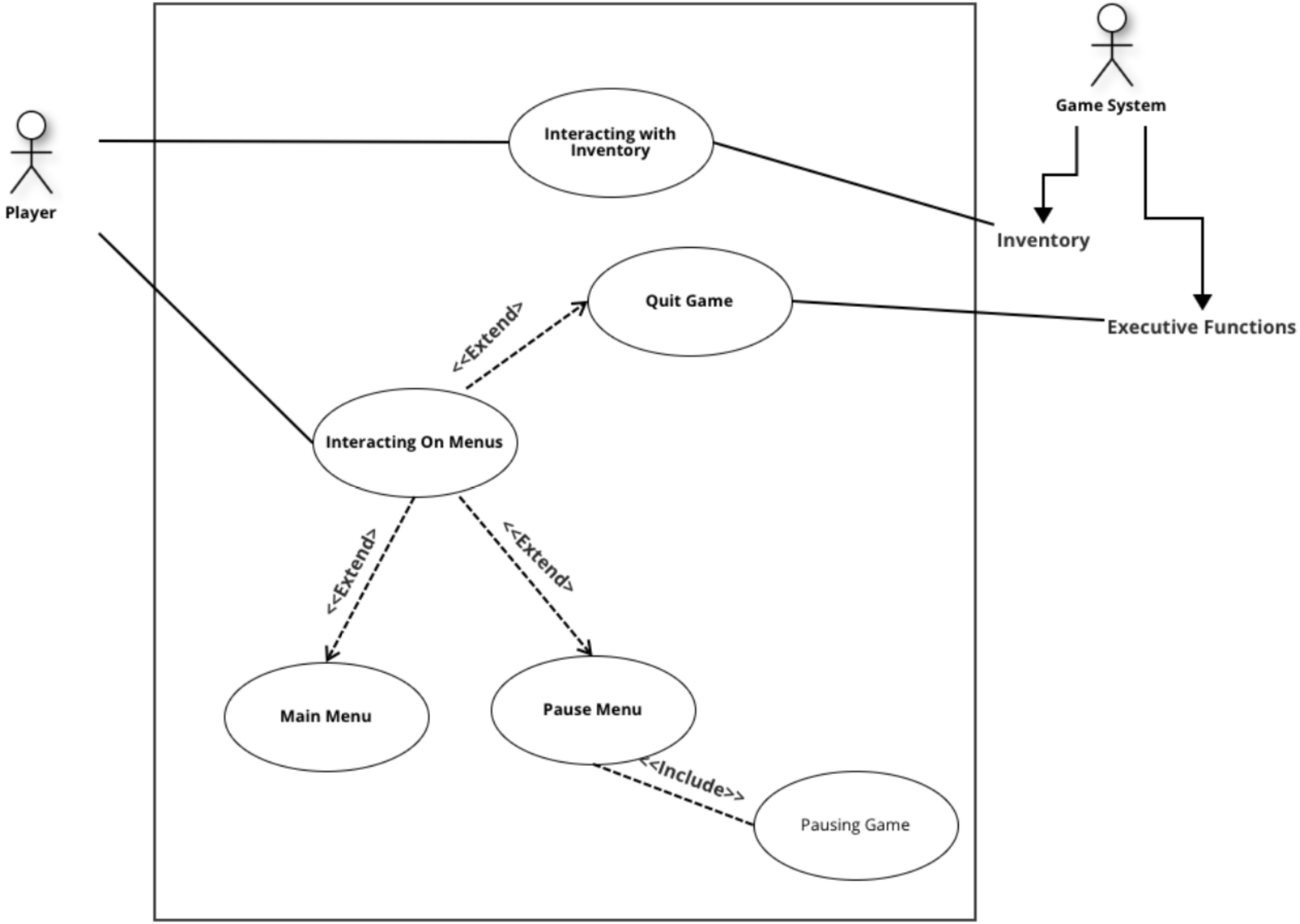
- Pick up items
- Use Consumables
- Drop / Place Items
- Display in HUD

Player

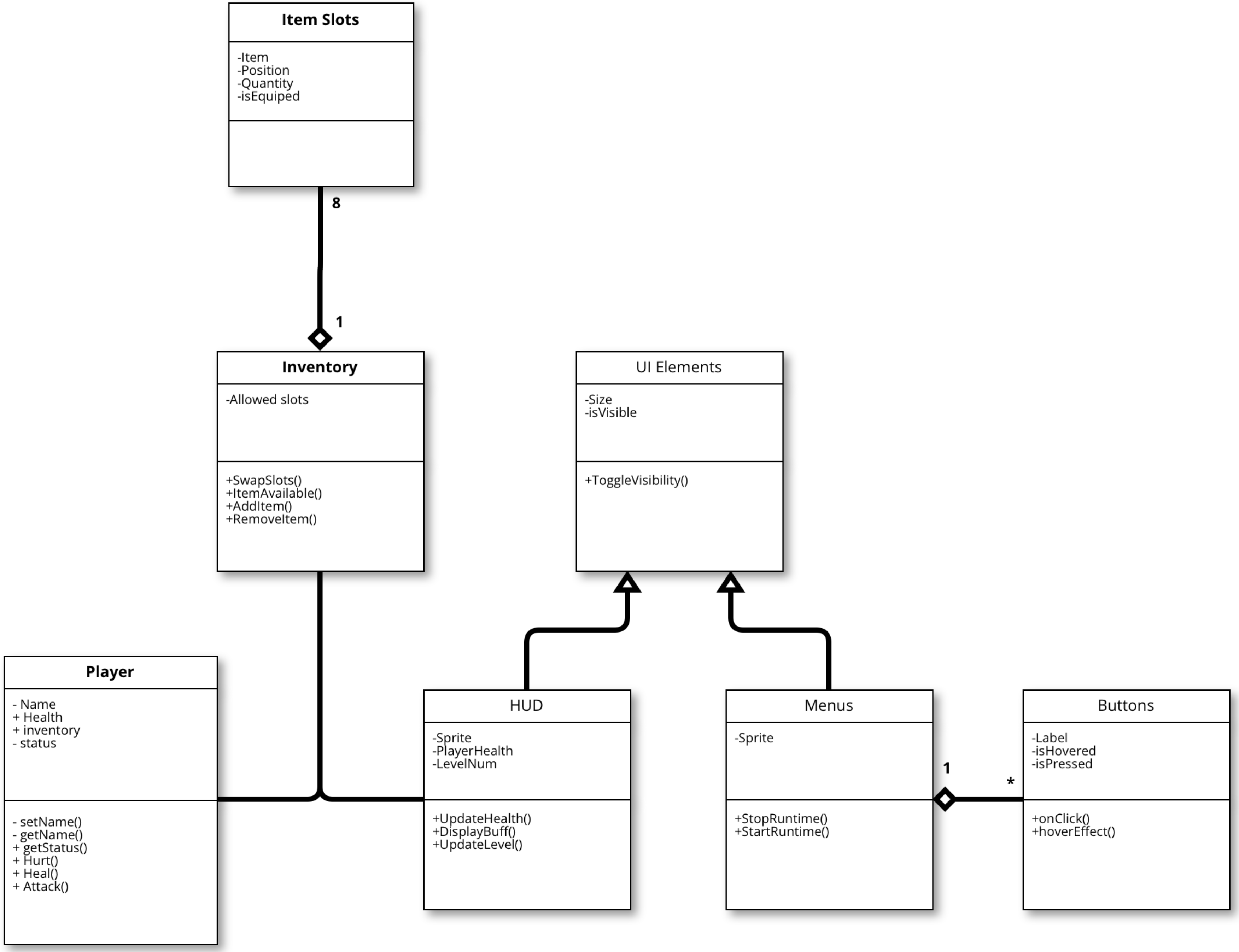
- Basic Movement
- Calling action functions from keybind
- Updating Player Statistics (health …)

Priority: 1
Complexity: medium

Collin – Use Case



Collin – Class Diagram



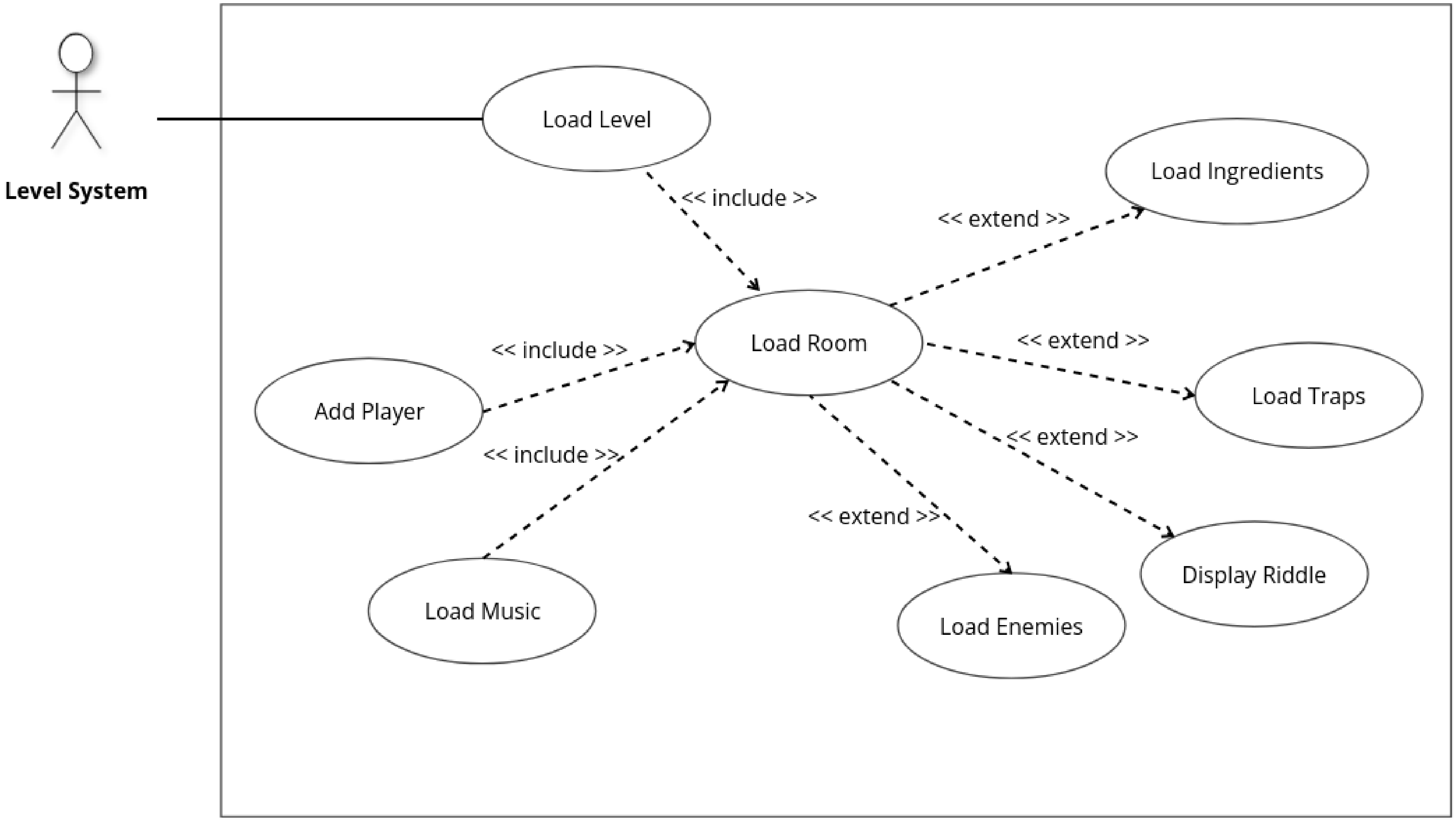
Scott – Level System Manager

Level Manager:

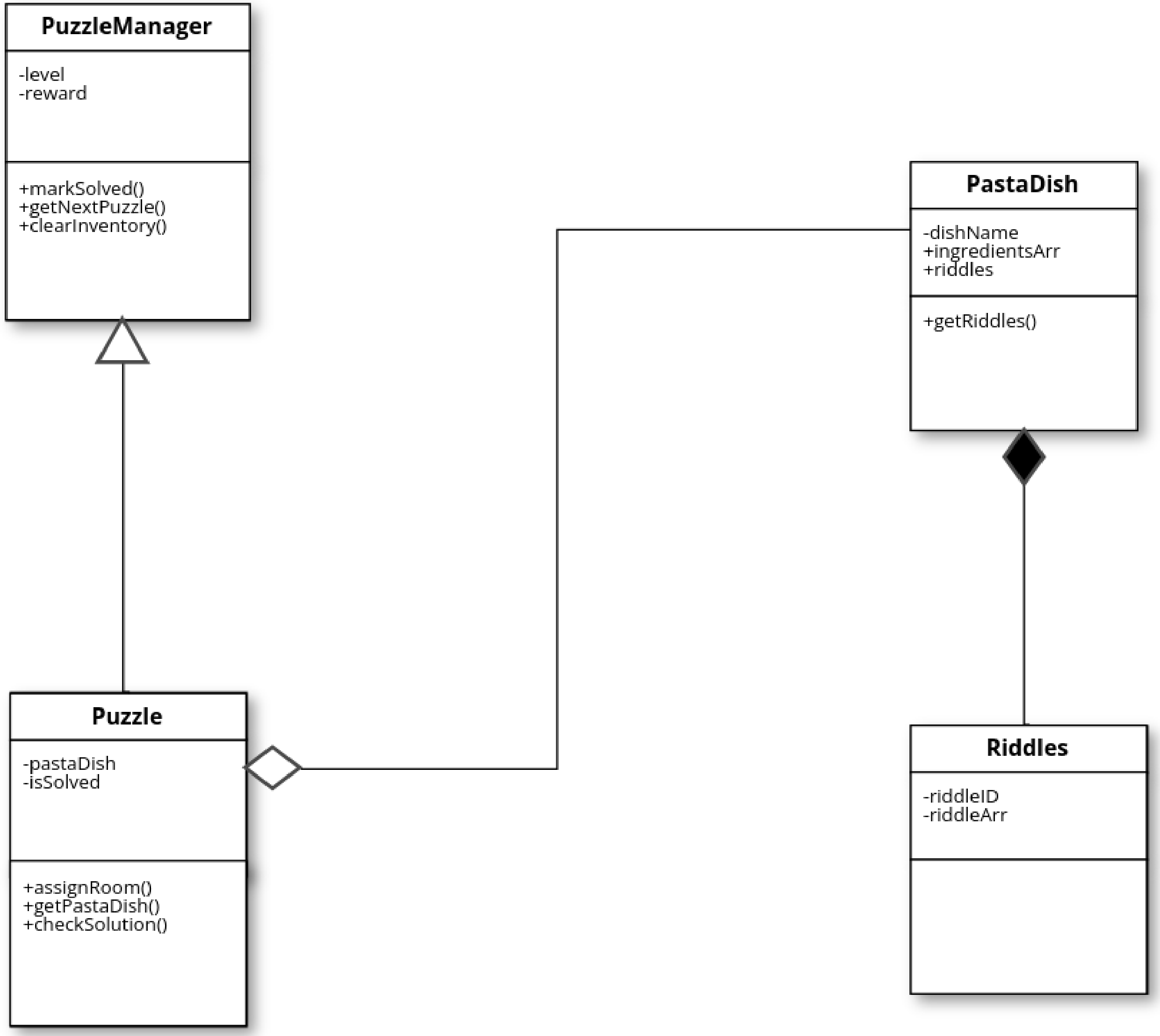
- Select the puzzle for the level
- Assign spawn locations for ingredients, enemies, and traps
- Load rooms that are being entered
    - Main room – Load main room scene, player, and Flying Spaghetti Monster
    - Riddle rooms – Load side rooms, player, enemies, traps, ingredients, and the correct riddle
- Play music
    - Main room – calm, soft music
    - Riddle rooms
        - While enemies are alive and an ingredient hasn't been selected, play action music
        - Once enemies have been defeated and an ingredient has been picked up, play soft calm music

Priority: 1
Complexity: Medium

Scott – Level System Manager
Use Case



Scott – Level System Manager
Class Diagram



PuzzleManager

-level
-reward

+markSolved()
+getNextPuzzle()
+clearInventory()

PastaDish

-dishName
+ingredientsArr
+riddles

+getRiddles()

Puzzle

-pastaDish
-isSolved

+assignRoom()
+getPastaDish()
+checkSolution()

Riddles

-riddleID
-riddleArr

Korben - Room Interaction

Ingredient pickups:

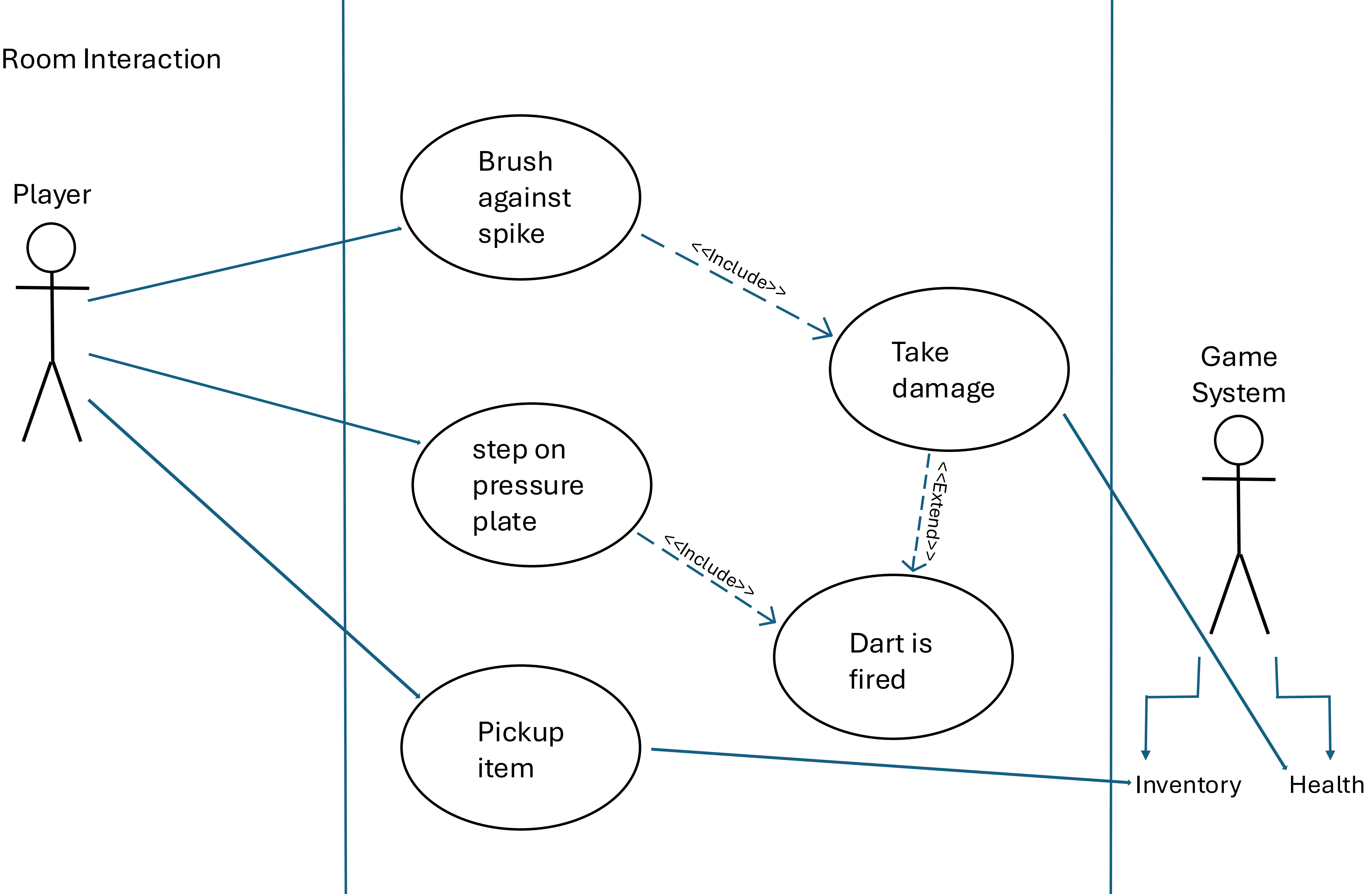
- Removing and moving the ingredients
- Making sure ingredients trigger an inventory update

Traps:

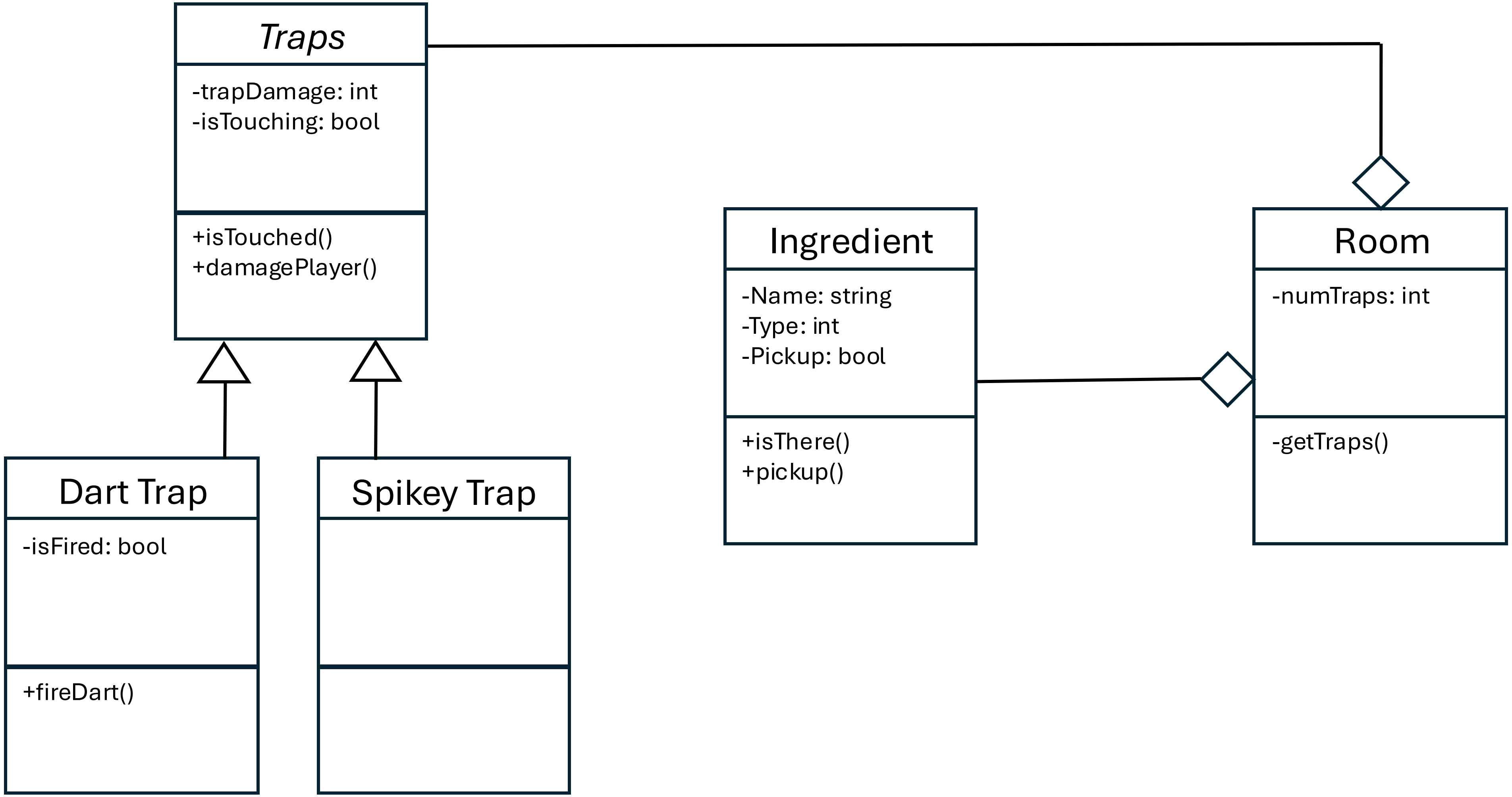
- Traps interact with heart bar to cause damage
- Shooting traps can be activated by pressure plates
- Stationary "spikey" traps deal damage on contact

Priority: 1
Complexity: medium

Korben- Room Interaction



Korben- Class Diagram



Traps

-trapDamage: int
-isTouching: bool

+isTouched()
+damagePlayer()

Dart Trap

-isFired: bool

+fireDart()

Spikey Trap

Ingredient

-Name: string
-Type: int
-Pickup: bool

+isThere()
+pickup()

Room

-numTraps: int

-getTraps()

Ben – Enemies / Demo Mode

Enemies

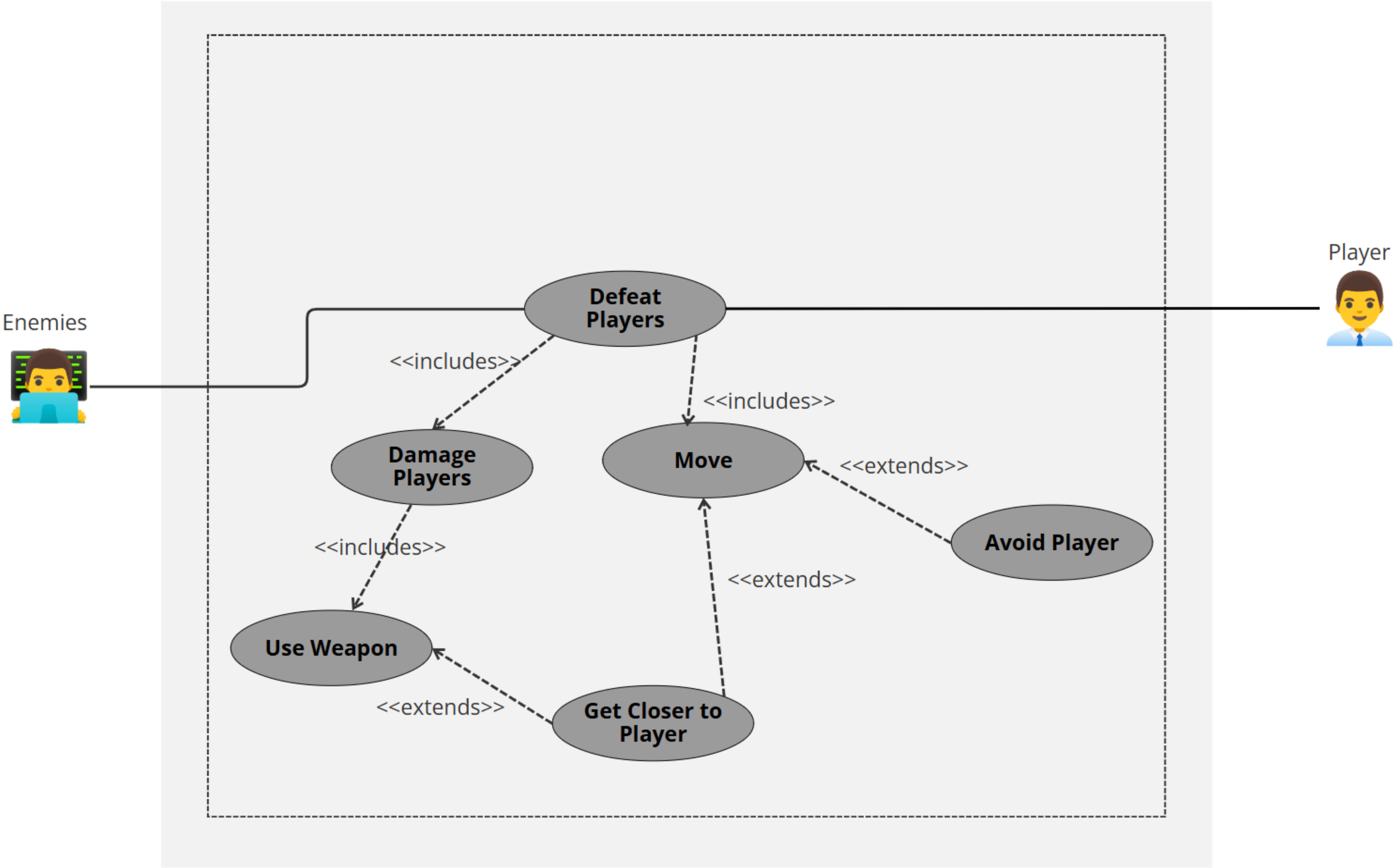
- Load Enemies
- Target Player
- Update Enemy Statistics

Demo Mode

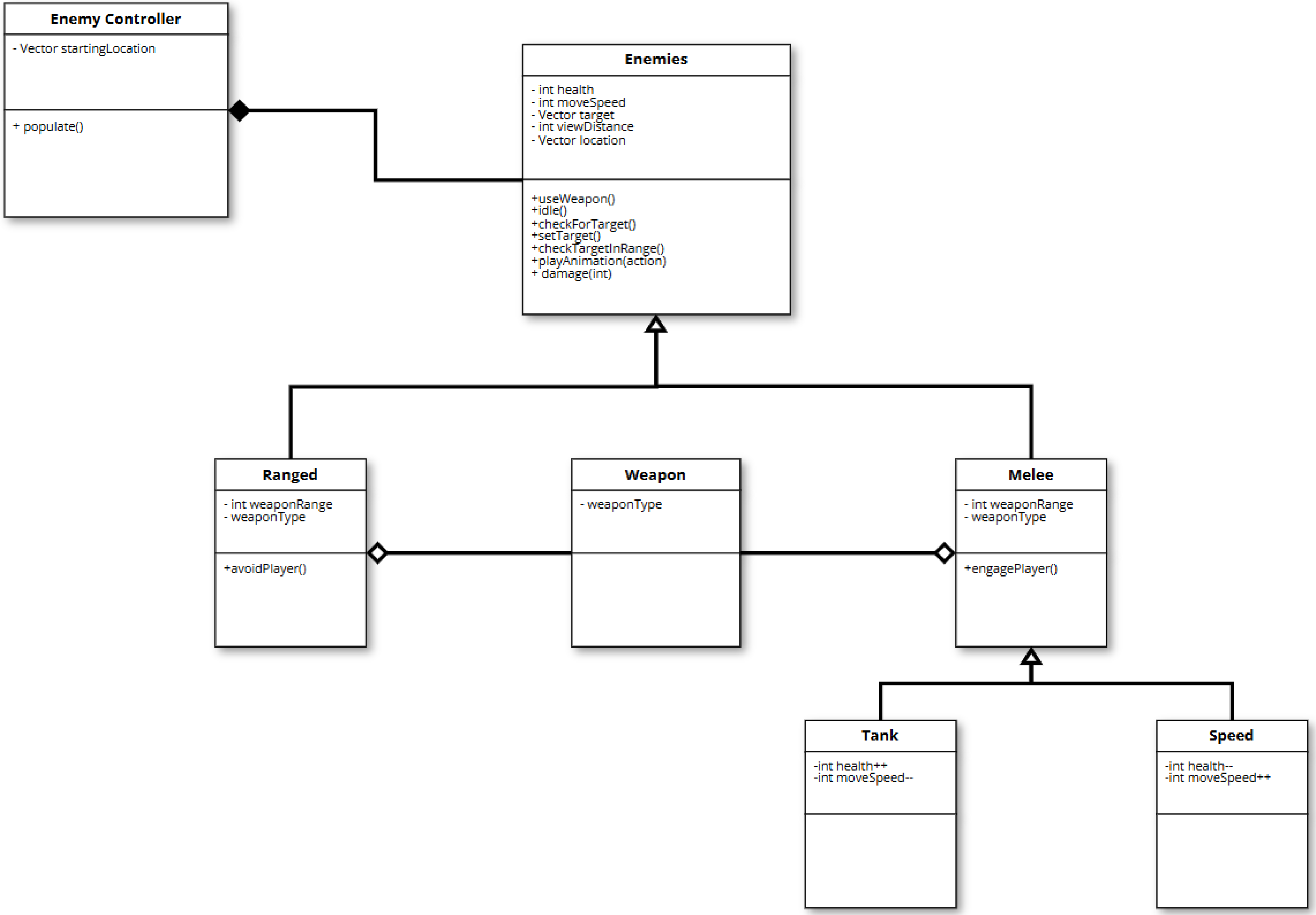
- Control Player
- Demonstrate Game Play
- Allow User to take control

Priority: 2
Complexity: hard

Ben – Use Case


Enemies
Player
Defeat Players
<<includes>>
<<includes>>
Damage Players
Move
<<extends>>
Avoid Player
<<includes>>
<<extends>>
Use Weapon
<<extends>>
Get Closer to Player

Ben - Class Diagram



Enemy Controller

- Vector startingLocation

+ populate()

Enemies

- int health
- int moveSpeed
- Vector target
- int viewDistance
- Vector location

+useWeapon()
+idle()
+checkForTarget()
+setTarget()
+checkTargetInRange()
+playAnimation(action)
+ damage(int)

Ranged

- int weaponRange
- weaponType

+avoidPlayer()

Weapon

- weaponType

Melee

- int weaponRange
- weaponType

+engagePlayer()

Tank

-int health++
-int moveSpeed--

Speed

-int health--
-int moveSpeed++

Garrett – Weapons

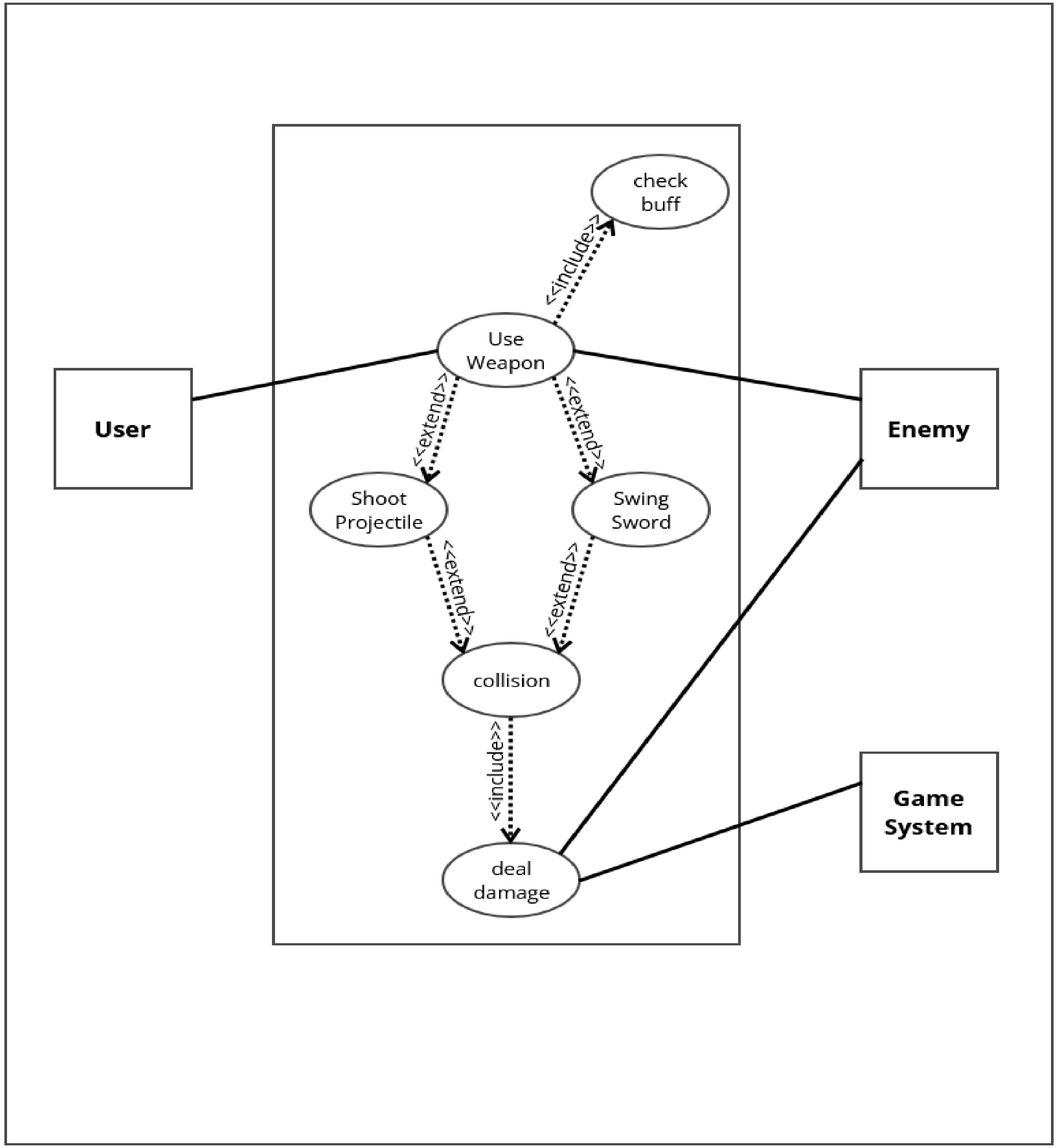
- Two different weapon classes
  - Melee Weapons
    - Two different melee weapons one sweep and one poke
  - Ranged Weapons

  Both User and Enemy can use weapon
  - Aiming class will take the cursor location and fire in that direction
  - Check buff class will also check for any active buffs that the user currently has

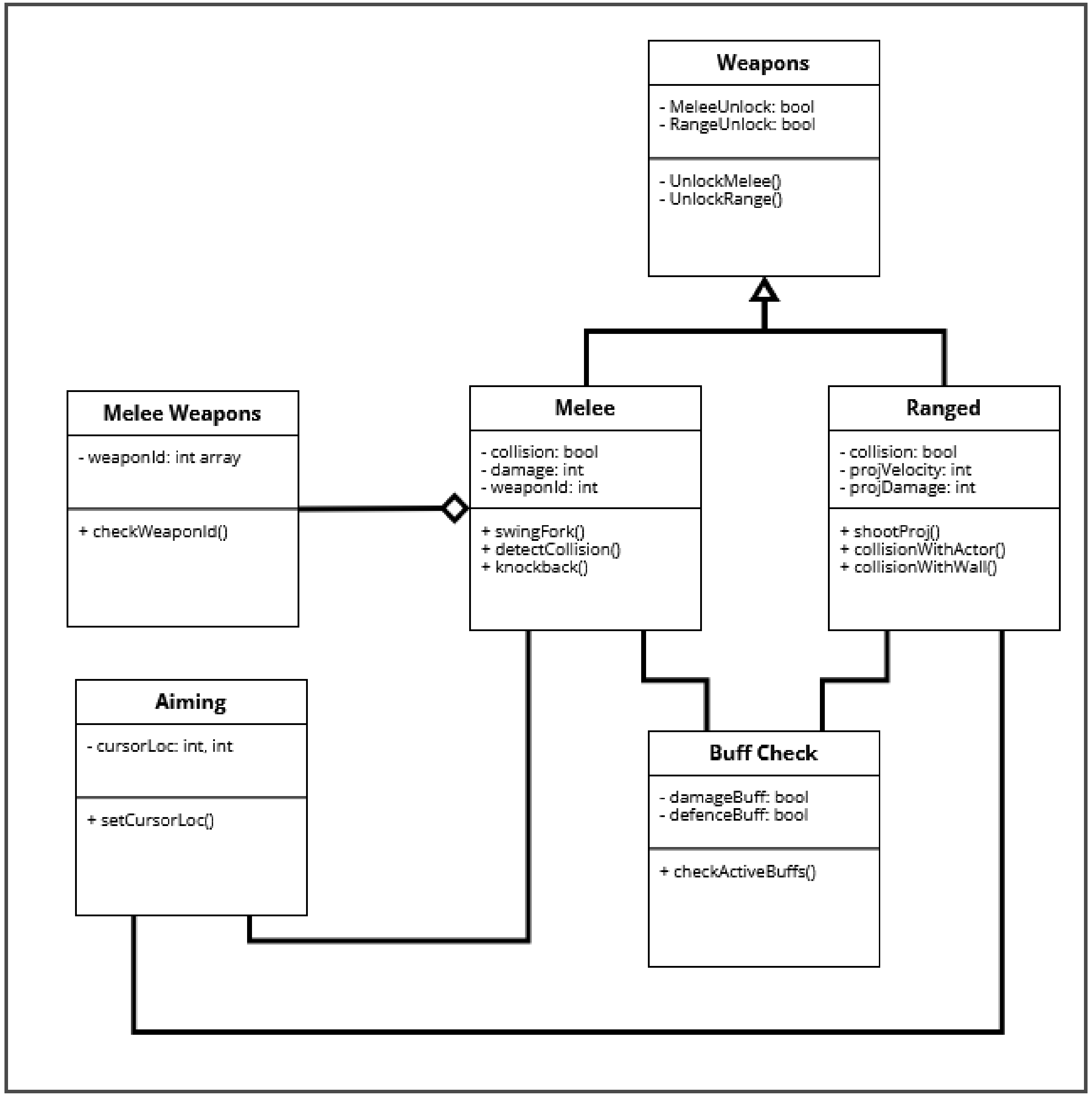
Priority: 1

Complexity: medium

Garrett – Use Case



Garrett – Class Diagram



Cole – Items

Item System

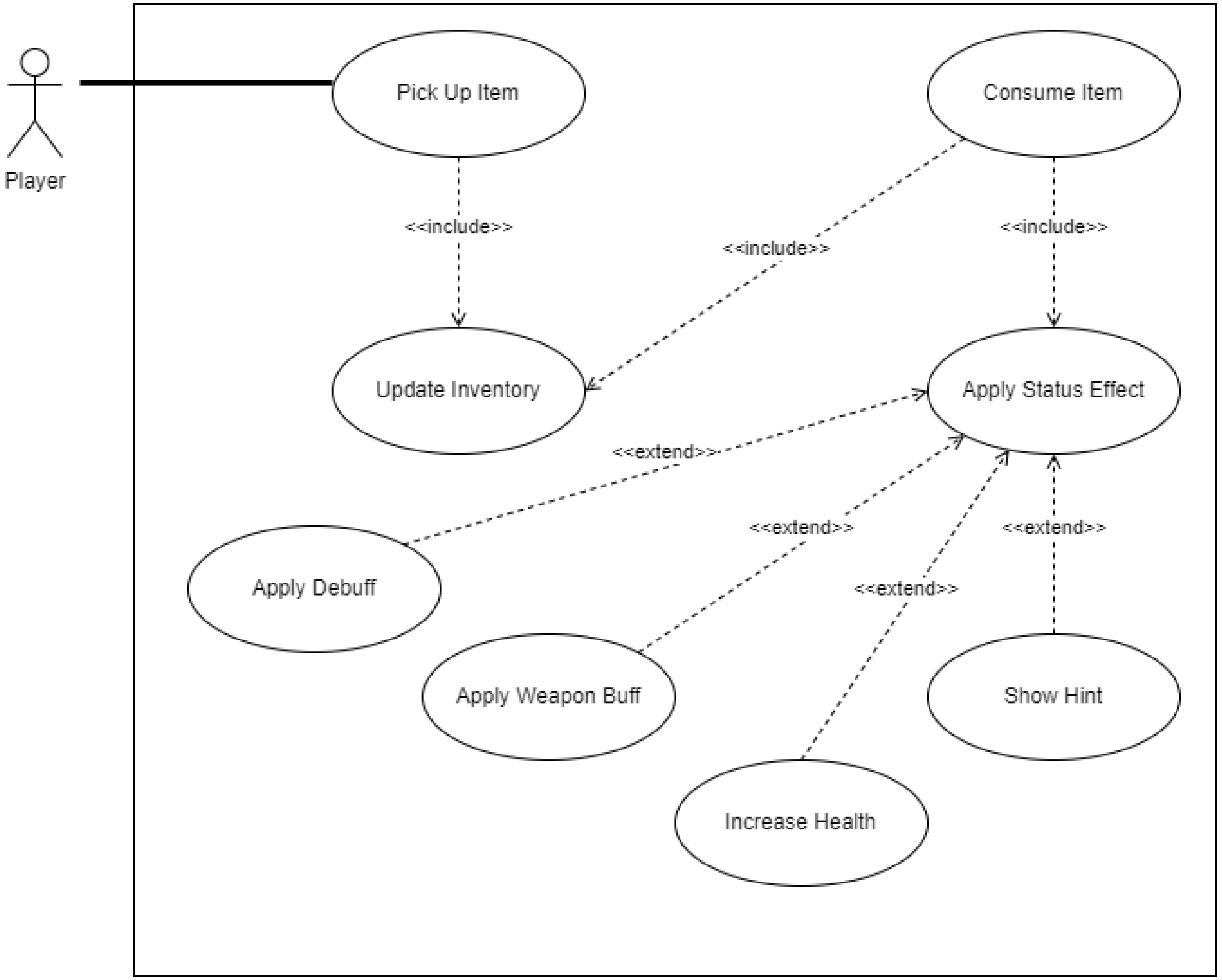
- Load consumable item into room
- When picked up update inventory
- When consumed apply a status effect and remove item

Power-Up system

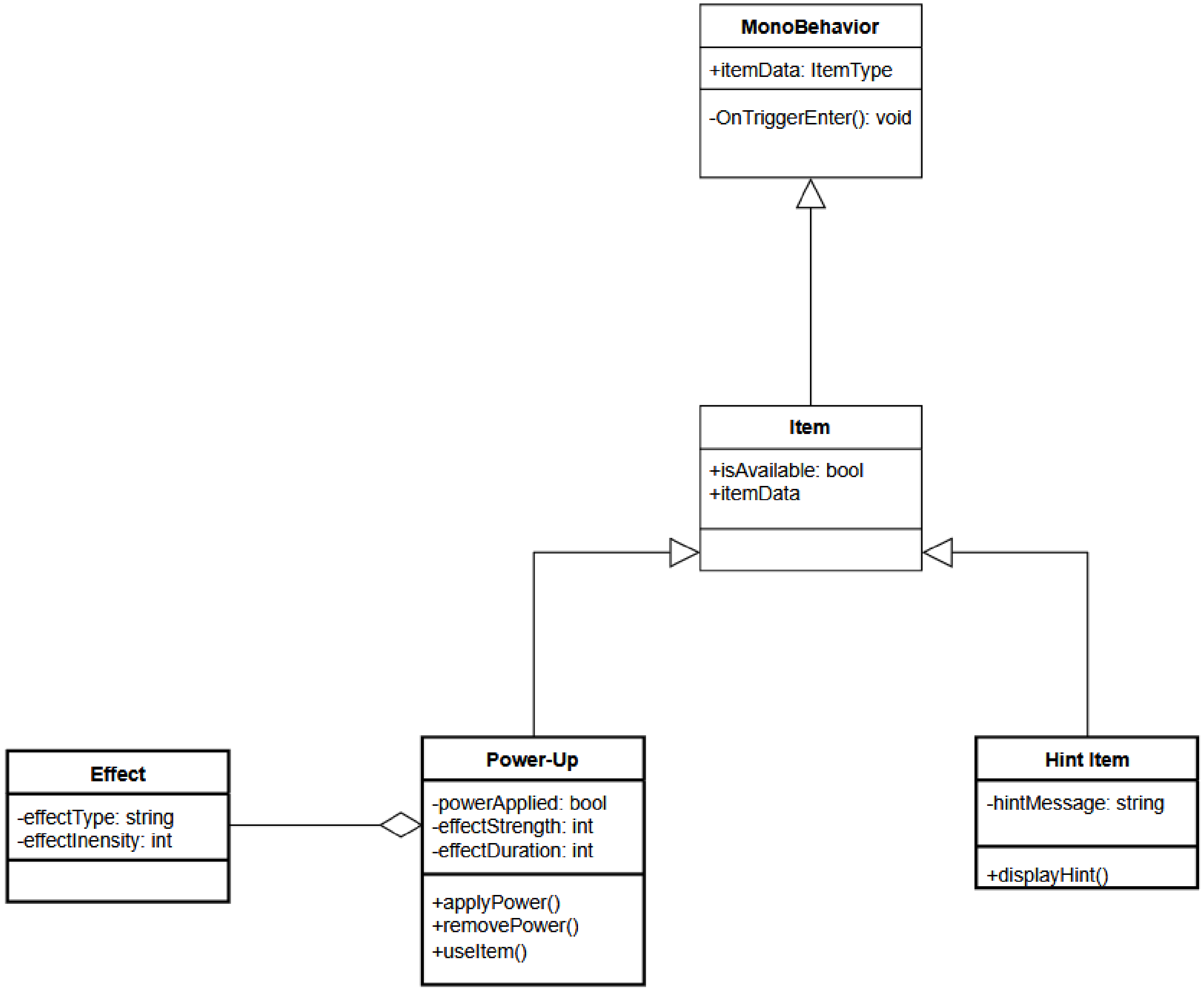
- When consumed apply an effect:
    - Weapon buff
    - Increase player health
    - Apply player debuff
    - Display hint

Priority: 2
Complexity: Medium

Cole – Items Use Case



Cole – Items Class Diagram



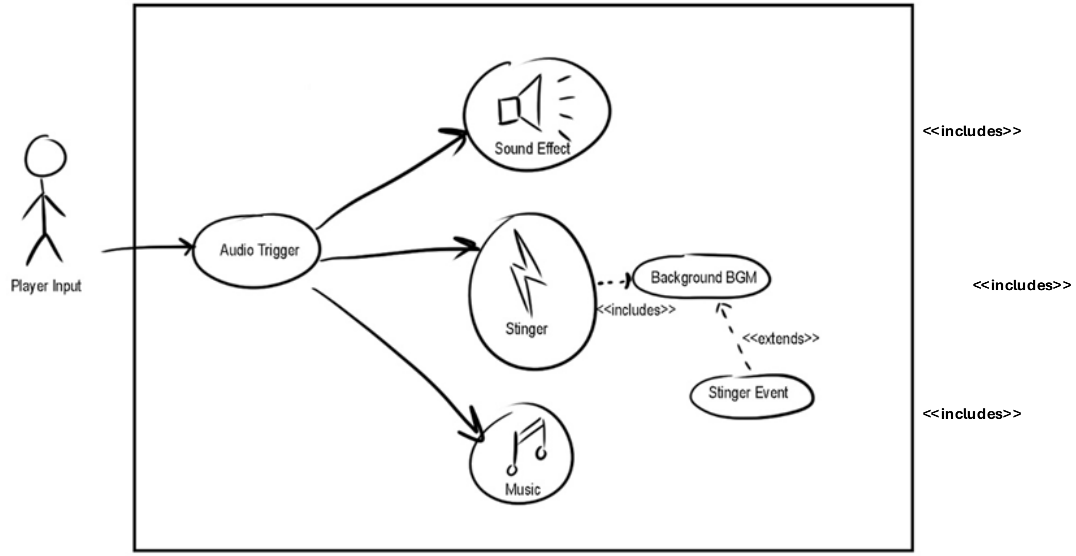
Jeffrey – Sound/Art

- Sound System
  - Music
    - Action Track when loading into hostile room
    - Once all hostiles on a level are dead, play chill music
  - Weapon/Item SFX
    - Enemies using their weapons/Player using their weapons
    - Sound upon damage for entities
    - When placing/using items, play sound
  - UI SFX
    - Play upon Click of buttons

Sound:
Priority: 2
Complexity: Easy/Med

Art:
Priority: 1
Complexity: Medium

Jeffrey – Use Case/Class Diagram



```
        +--------------------------------------+
        |          SoundEngine (S)             |   <== Singleton
        |--------------------------------------|
        | - instance: SoundEngine (static)     |   (private static)
        | - soundEffects: List<Sound>          |   (aggregate collection)
        | - musicTracks: List<Music>           |   (aggregate collection)
        | - eventTriggers: Map<string, Sound>  |   (event-driven triggers)
        |--------------------------------------|
        | + getInstance(): SoundEngine         |   (static getter)
        | + playSound(name: string)            |
        | + playMusic(track: string)           |
        | + stopMusic()                        |
        | + fadeMusic(duration: float)         |   (volume fading)
        | + registerListener(l: SoundListener) |
        | + notifyListeners(event: string)     |
        | + triggerEvent(event: string)        |   (stinger trigger)
        +--------------------------------------+
                           |
                           |  <<Inherits>>
                           v
        +--------------------------------------+
        |      SoundManager : MonoBehaviour    |  (Manages Unity Integration)
        |--------------------------------------|
        | + loadSounds()                       |
        | + updateVolume(level: float)         |
        | + fadeMusic(duration: float)         |
        | + handleGameEvent(event: string)     |   (React to in-game events)
        +--------------------------------------+


Aggregates
+---------------------+       +---------------------+
|     Sound (Private) |       |   Music (Private)   |
|---------------------|       |---------------------|
| - name: string      |       | - track: string     |
| - clip: Audio       |       | - bgm: Audio        |
+---------------------+       +---------------------+


            Implements Observer Pattern
                           |
                           v
        +--------------------------------------+
        |          SoundListener (I)           |   <== Interface
        |--------------------------------------|
        | + onSoundPlayed(event: string)       |
        +--------------------------------------+
                           |
                           v
        +--------------------------------------+
        |   UIAudioHandler : SoundListener     |   (Example Implementation)
        |--------------------------------------|
        | + onSoundPlayed(event: string)       |
        | + adjustUIVolume(level: float)       |
        +--------------------------------------+


            Event-Driven Triggers
+-------------------------+       +-------------------------+
|  StingerEvent (Private)|       |  VolumeEvent (Private) |
|-------------------------|       |-------------------------|
| - trigger: string       |       | - effect: string        |
| - sound: Sound          |       | - volumeChange: float   |
|-------------------------|       |-------------------------|
| + playStinger()         |       | + applyVolumeChange()   |
+-------------------------+       +-------------------------+
```

```
  **Possible Event Interactions:**
  --------------------------------
  - **Puzzle Completed** -> `SoundEngine.triggerEvent("puzzle_complete")`
      - Plays Stinger sound
      - Fades background music momentarily

  - **Critical Hit** -> `SoundEngine.triggerEvent("critical_hit")`
      - Boosts volume temporarily for impact


  - **Enemy Appears** -> `SoundEngine.triggerEvent("enemy_alert")`
      - Triggers suspenseful stinger
      - Lowers background music for tension

  - **Low Health Warning** -> `SoundEngine.triggerEvent("low_health")`
      - Gradually fades volume and adds heartbeat effect
```

Questions?