Kayden Adams

Homework 2 - Project Proposal

1. I will be making a rouge-like dungeon crawler. It will follow the traditional dungeon crawler loop of enter dungeon -> fight monsters and collect loot until you die -> get rewards for progress (I will do this in the form of only slightly lowering your player level, which increases health, attack, and defense) -> enter dungeon with benefits and hopefully get further. The dungeon will be grid based, so players can move one tile at a time, and it will be turn-based - for every one move the player makes, the monsters also make one move. The player will move using the arrow keys and attack using the Z key. There will be no diagonal movement, only up, left, right, and down. They will be able to change direction in place by using shift + arrow keys, which will not take a turn. If they attempt to run into a wall or into a monster, they will not move and a turn will not pass, but they will change direction. The dungeon will be procedurally generated with several rooms and hallways between them. There will be multiple types of monster which will wander randomly from room to room. If they see the player, they will move towards the player and then attack them. I think that I only want to do close range enemies who have to be next to the player to attack and not deal with ranged enemies firing projectiles and trying to get further from the player but in shooting range. That would require an entirely separate AI, additional sprites, and game balance issues that feel like too much for 10 weeks. I am unsure if I want to include an inventory and items, but I have sprites for them if I decide to do some. When the player attacks, they will attack directly in front of them. Damage dealt will be based on the player's attack and the monsters defense, with a minimum number of damage of the attackers level*2. Monsters attacking will work the same. If a player attacks, regardless of if they are attacking a monster or not, a turn will still pass.

2. I'm going to use a scale for familiarity of 1-5 for familiarity, with 1 being "I've never used this before." and 5 being "I've used this frequently and I'm very comfortable with it." Pygame and Python Arcade are the two game development frameworks I've been considering, but I would one of them and not use both. I wanted to get advice from the professors and do more research into both before my absolute final decision, but I'm currently leaning more towards arcade because of this comparison. Arcade also has detailed tutorials and sample code for a procedural generated dungeon, grid based games, and a number of other important aspects (like moving using arrow keys or switching animations) Additionally, Python Arcade seems to have better documentation, so I'm going to act as if I am using Python Arcade with the ability to switch to Pygame. I found a dungeon built in Python Arcade here and one built in Pygame here so I have proof-of-concept that these frameworks will support the type of game I want to build. With all that said, I think will be using the following technologies:

Technology	Used For	Familiarity
Python	OO coding language	5
Python Arcade	Game development framework and GUI	1
Git and Github	Version Control	4
Travis CI	Continuous Integration	2
Pytest	Testing	3

3. My project won't work if it doesn't have:
    a. A character for the player to move. This one is non-negotiable. If I can't get this piece working, I need to completely change my plan. Fortunately, it should be fairly simple. I've looked over code on Arcade and moving the player using the arrow keys seems fairly easy. There is even demo code the creators of Arcade have already written to move a player, although not on a grid-base system.
    b. Monsters who roam around. This one is also pretty essential. The AI is flexible, I want to create a sort of "field of view" where the monsters attack the player if they can see them, and smart decisions where a monster will proceed down a hallway and go from a room to another hallway. But if I can't do that, I can just make a basic path-finding algorithm to have the monsters just go straight for the player.
    c. Attacking functionality. Again, important for a dungeon. I'm willing to do away with stats like Attack and Defense if need be and just have things do a flat amount of damage.
    d. A "camera" that follows the player. I want to only show some of the map and have what the player sees change as they move. If I can't get this working, I will show the entire map to the player, maybe with graying out areas they can't currently see.
    e. Sprites. If I can't get the sprites that I have found to work, I am able to just draw basic shapes for monsters, the player, and the world. I am planning to start with just different colors of squares and expand from there.
    f. Procedurally generated dungeon map. I could always just hard-code some different floors if need be, or simplify the generating from creating rooms and hallways to just making one big room and randomly placing walls.
    g. Multiple floors. This gives a sense of progress to the game and increases difficulty. It isn't strictly necessary, as I could just spawn monsters in waves of increasing difficulty, but it would be nice to have.
    h. A UI to track damage and progress. In its most basic form, this could just be a box with text output like "You dealt 5 damage. The monster dies! You level up!" but ideally I would love to have a health bar and EXP bar with stats for attack and

defense, plus a number that pops up over a sprite when they take damage. Maybe both if I'm able.

4. The main outside resource I need is artwork. I will be using Spriter's Resource which I've used a bit before. It provides sprite sheets from thousands of different games. These include characters with animations for idle, walking, running, attacking, using items, and other actions. It also has tilemaps for different floors as well as item sprites. I will need a main character with walking, attacking, and item use animations, at least 3 enemies with walking and attacking animations. I also need a tile map I can use to create dungeon floors. Currently, I'm thinking of using sprites for monsters and the main character from here because they are fairly small and evenly sized, and looking over the sprite sheets they all have up right left down and attack animations. I think I will use this sheet for the main character and this, this, and this for the monsters. That game doesn't have a tile set, so I think I will use this one, because it is a similar style and looks thematically appropriate for a dungeon crawler. If I find it doesn't work well, Pokemon Mystery Dungeon is a dungeon crawler and there are an enormous amount of tile sets for it (bottom of the page) If I use them, I will use item sprites (like food or weapons) from here. I know a little bit about cutting out sprites and creating animations from them, but I also have a friend who is an artist who has offered to help, either cutting out sprites or making ones if I can't find them. Other than artwork, I also need some kind of soundtrack, and if I have time, sound effects. I'll probably pull the music from here and the sound effects from here. I believe Pygame has some sound effects built in as well if I choose to use it.

5. Architecture below

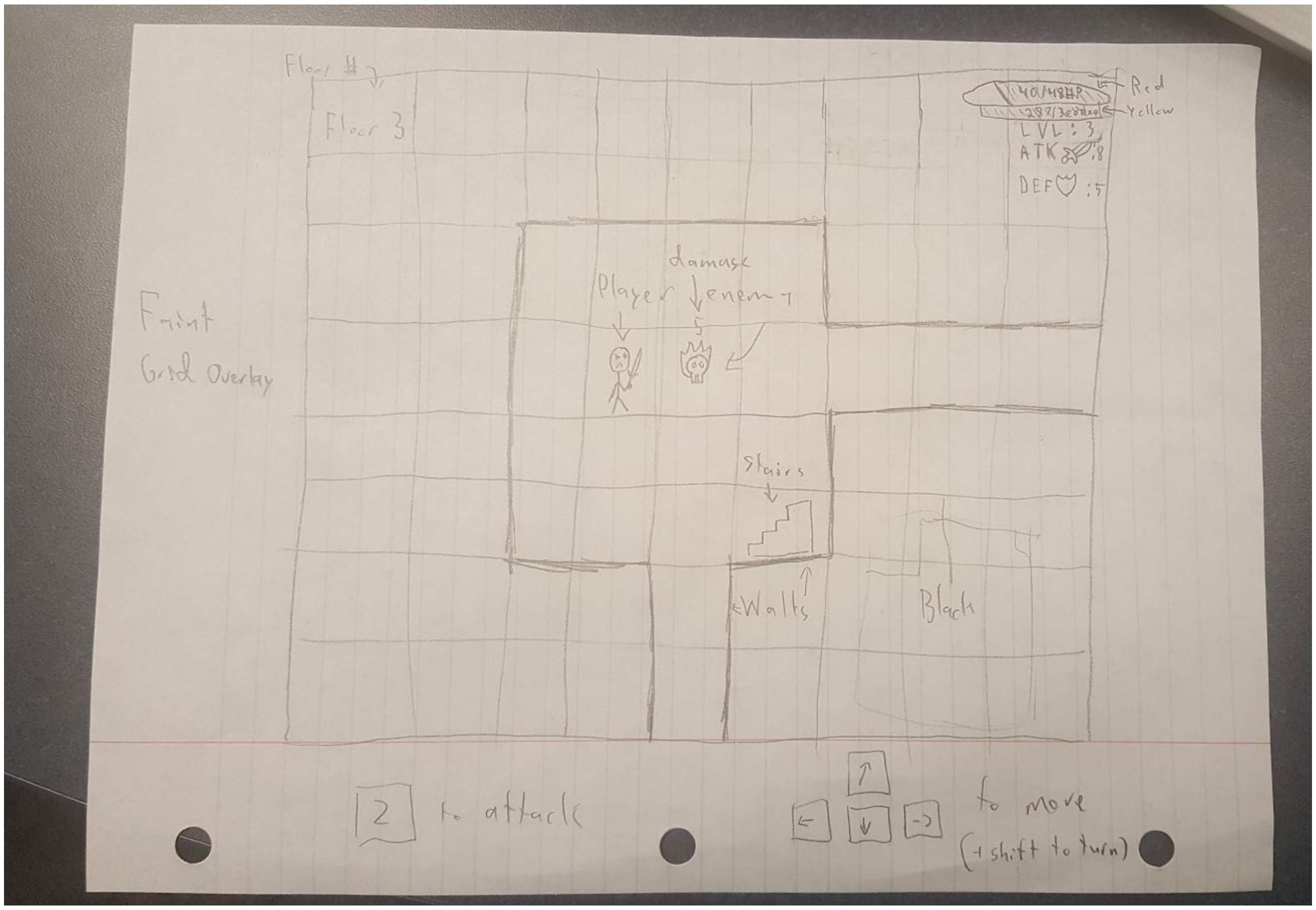
Class Name	**Type**	**Details**	**Interaction**
Mover	Interface for Monster and Player	Mover is an interface the Player and Monsters will use an instance of to move around the board. They both move in the exact same way, the only difference is the Player's movement is decided by by the arrow keys and the Monster's movement is decided by AI.	Monster and Player will both have an instance of this and use it.

Attacker	Interface for Monster and Player	Attacker is similar to Mover. It is what will deal with a Monster attacking a Player or vice versa. It will calculate damage dealt and be used when an attack happens.	Monster and Player will both have an instance of this and use it.
Animator	Interface for Monster and Player	Animator is what will give animations for Monsters and Players. Animator will be what sets monster and player animations to different things depending on their action (like moving) and their direction (like left)	Monster and Player will both have an instance of this and use it.
Monster	Abstract super class for enemies	Monster will include instances of Mover Attacker and Animator for drawing, moving, and attacking that should be universal among the different monsters. This will include the AI that drives the monsters to move closer to the player and attack them. It will also have setters and getters for the attributes, which will be Level, Attack, Defense, ExpValue, and Position. It will define an __init__ will take 1 parameter, level. The subclasses will do the implementation for this where they will and scale up attack, defense, and exp by the level given. Each subclass has a different base attack, defense, and exp. Monster will use Animator, Mover, and Attacker to do actions	Monsters will be created by MonsterSpawner
SkullMonster	Subclass of Monster	Skulls will have higher attack and lower defense.	Skull is a subclass of monster
FishMonster	Subclass of Monster	Fish will have equal attack and defense.	Fish is a subclass of monster

LampMonster	Subclass of Monster	Lamps will have lower attack and higher defense.	Lamp is a subclass of monster
MonsterSpawner	Factory for creating monsters	MonsterSpawner will return instances of Monsters at different levels. If I implement items, I will upgrade it to an Abstract Factory that builds the monsters with items that you get when you defeat them.	MonsterSpawner will return instances of Monster based on the string that MakeMonster is given.
DungeonMap	Map of the dungeon	Dungeon Map is used sort of like a very fancy version of the Board for homework 1. It will track the tilemap of the dungeon, including the walls and the floor tiles. It will also track the location of the sprites of the player and monsters.	DungeonMap will have an instance of MonsterSpawner that it uses to create a monster. Then it will place the monsters within the map.
GenerateDungeon	Procedural Generator	GenerateDungeon will procedurally generate a dungeon map. This will be used once, to create the map. This is just used to generate the tiles for the map, not to place monsters or the player.	GenerateDungeon will make a DungeonMap

Player	Singleton Player character class	Player is a singleton because there should never be more than one instance of the player. Player will track a Player object. It will have attributes of FullHP, CurrHP, Level, Attack, Defense and animations for moving, standing, and attacking left, right, up, and down that Animator will use. It will have getters and setters for its attributes and use Animator, Mover, and Attacker to do actions.	Player will be using Attacker Mover, and Animator
MyGame	Main Runner Class	MyGame is what Arcade uses as its main runner so that's what I'm going to call it. It deals with keypresses, drawing the screen, updating attributes (through the getters and setters Player and Monster have), dealing damage (using Attacker to calculate), everything the game needs to run properly.	MyGame will call GenerateDungeon during setup to make a dungeon floor. It will save the resulting DungeonMap. It also has a reference to a Player and a list of Monsters. It will be using Attacker, Animator, and Mover through the Monster and Player classes. It may also have an Animator depending on if that functionality makes sense (Like I need to be animating water or a special effect like a fireball) This will be the main thing passing information between the DungeonMap, the Player, and the Monsters/MonsterSpawner.

6. The good news is that Arcade takes care of most of the GUI setup and provides methods for drawing and updating. I've included a picture of what the GUI will look like.



7. See below

Date	Summary	What Will Be Done	Knowledge Needed	Learned
2/21/20	Basics	Basic foundations. I will have a Github repo with an Arcade file. When run, the Arcade file will open Arcade's GUI and display a blank screen (code for this is on Arcade's website) This Github will be linked to TravisCI. I will have a test file with a mock Pytest test that TravisCI will run.	Python, Github, TravisCI, Pytest (All following will also need these)	I'm very unfamiliar with TravicCI, though I've technically used it before. That will be the main learning point for this week.

2/28/20	Definitions for classes and basic map	All Classes will have the definitions for their methods and attributes defined (I will do the Python equivalent of a header file) Largely, I see myself using this time to familiarize myself with Arcade. I will have created a grid-based map that is just a 5x5 square surrounded by walls. I will be using simple squares for this, not cutting out sprites or creating animations.	Python header files/class definitions, Arcade usage, Grid-based Arcade	Python header files/class definitions, Arcade usage, Grid-based Arcade
3/6/20	Movable player	I will have completed and tested Mover. Additionally, I will have a Player, still represented by a square that the human player will be able to move around the tile room in a grid-locked manner by using the arrow keys.	Keypress detection in Arcade, Arcade movement, Arcade grid locking movement,	Keypress detection in Arcade, Arcade movement, Arcade grid locking movement,
3/13/20	Monsters and MonsterSpawner, Monsters randomly placed. AI that move towards the player only.	Monster and MonsterSpawner will be finished. Monsters will each have stats that are set depending on their level. MonsterSpawner will be returning instances of Monsters to the main runner, which will place them in a random valid location around the map. Monsters will move towards the player based on a simple Breadth First Search.	Factory Pattern, esp in Python, smart randomizatio n.	Factory Pattern, esp in Python, smart randomizatio n.
3/20/20	Attacker finished and player and monsters can attack and die, player can level up. Restart game if player dies.	The Attacker will be finished. Players and Monsters will be able to attack each other and be able to die. If the Player dies, there will be an option to restart the game. If a monster dies, the player should get EXP and be able to level up.	Arcade changing attributes, game balance issues, Arcade restarting a game on death, Python deletion of objects.	Arcade changing attributes, game balance issues, Arcade restarting a game on death, Python deletion of objects.

3/27/20	Procedural dungeon generation	This is the big one. The next two weeks will be spent on procedural generator for the grid-based dungeon. The first week will largely be learning about how procedural generation works, then getting started on it and getting as far as I can.	Procedural generation, esp grid-based.	Procedural generation, esp grid-based.
4/3/20	Procedural dungeon generation cont now with different floors and camera that follows player	More dungeon generation! I should have a pretty good handle of how to go about this from last week, and hopefully I have made some progress. By the end of this week, I should have a completed GenerateDungeon. The player should be able to advance to a new floor by stepping on a staircase. A camera should follow the player around so they can only see some of the map.	Procedural generation, esp grid-based. Camera controlling in Arcade, Changing Level in Arcade	Procedural generation, esp grid-based. Camera controlling in Arcade, Changing Level in Arcade
4/10/20	Sprites all cut out and linked to their things. Walls and floors replaced with tile set.	Time to make the game look good and not just cubes. All of the spirits will be cut out and ready to be linked to their objects or their objects Animator. The wall and floor tiles will all be changed from cubes to sprites.	Cutting sprites, sprite linking in Arcade	Cutting sprites, sprite linking in Arcade
4/17/20	Animator finished - player and monsters have animations for moving and attacking.	I will have finished the Animator Class. This means that Players and Monsters will all have animations when they are moving and attacking, and I think death animations should be doable too.	Animation in Arcade.	Animation in Arcade.

4/24/20	Complete UI.	Depending on how familiar I am with the software, this might come in the form of a little text window or it might be a health and stats bar and minor animations for numbers when damage is dealt. Either way, there will be a UI to show the player what is happening in the game and display important information like stats.	UI design, Displaying stats in Arcade, text-windows in Arcade	UI design, Displaying stats in Arcade, text-windows in Arcade
4/29/20	Add sound effects, music. Beef up AI.	Polishing time. This last bit is for squashing bugs and making the game really shine. This means adding sound effects and music, minor special effects, beefing up the enemy AI, and making sure everything is running really well.	Sound usage and design, sound effects, AI design, bug testing	Sound usage and design, sound effects, AI design, bug testing

8. I will continue to regularly attend class and participate in the in-class exercises. I will continue to get my Programming Exercises in on time. I will attend office hours if I feel that I need help or don't understand material in class. I have not had trouble staying engauged in courses in the past and I have always done well in them. I will ask for help if I need it from instructors and classmates. Even though I will not be using C++ or Qt in the class, I feel fairly comfortable in C++ and I believe my grades reflect this. I haven't had trouble keeping up so far and I was able to complete Homework 1 with very minimal assistance. I completed both the base assignment and the extra credit and got the Homework in before the due date by a decent amount. I will address anything that comes up through open communication with the course staff in a timely manner.

9. Stretch Goals/Optional Extensions:
    a. Spend Points: The Player could choose to increase Attack, Defense, or HP when they leveled up rather than auto-leveling. This would also likely lead to modifying other parts of the game to ensure it is balanced correctly.
    b. Boss Monster: A larger, more powerful monster on floor 10 that you have to beat to win the game. This monster might take 4 squares instead of 1, and could have attack patterns where it charged a strong attack and you would have to avoid it.
    c. Better Monster AI: Exactly what it sounds like. The monsters would have a more sophisticated AI. Some of them might run away if their health gets too low, maybe some could try to attack from a distance. They could have a varying "field

of view" where they could see a few squares in front of them in a hallway but see the whole room in a room or doorway.

d. Consumable Items: Items like health potions that you step on to increase your health, or a scroll you step on to damage everything around you. Maybe monsters would drop these sometimes.
e. Inventory: This one is a pretty big extension, but it would be cool. There could be an inventory where you kept consumable items in to use later, like scrolls or potions.
f. Equipable Items: Another big one, mostly due to the UI. There could be slots where you could equip items like a new weapon or armor. Monsters would drop these or maybe you would find them lying around sometimes and you could use them to give yourself an advantage.
g. Traps: Certain tiles are "trapped" with invisible traps that only become visible when you step on them. These could do something like deal damage to you, take away an item, temporarily lower a stat, or spawn a monster near you. They might also have a positive effect, like giving you EXP or restoring health.
h. Deployment: I could deploy the game using Heroku or Digital Ocean. I'm unsure if this is possible or not, but it would be something to look into.