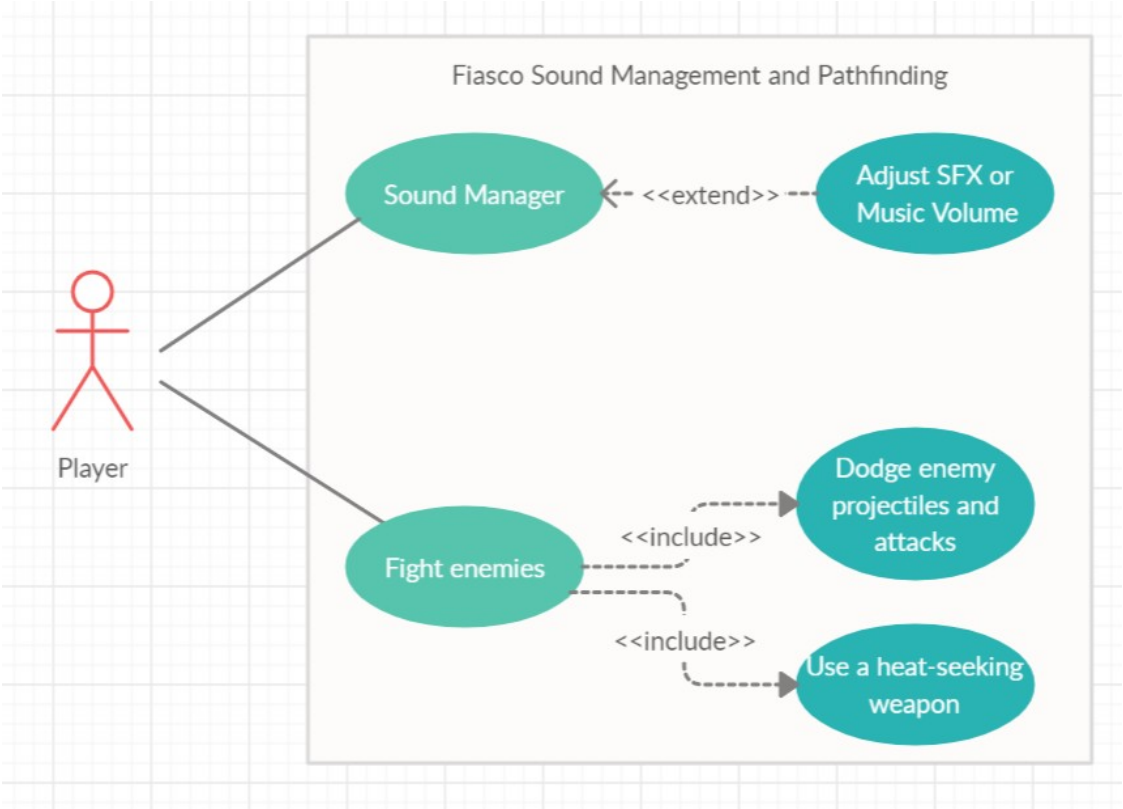
Name: Greyson Biggs Mark _____/50

1. Brief introduction _/3

My name is Greyson Biggs, and I'm Team Lead #6 in charge of Fiasco's Sound Manager, as well as Pathfinding. The Sound Manager accepts signals triggered by certain in-game actions, and plays corresponding sound files. Sound volume can also be adjusted in the settings menu, which the sound manager then handles accordingly.

The Pathfinding System is assisted by an external library (the Astar Pathfinding Project) and plays a significant role in dictating enemy behavior and the behavior of certain projectiles (e.g. heat-seeking missiles, whether fired by the player or an enemy).

2. Use case diagram with scenario _14



Scenarios

Name: Adjust SFX/Music Volume

Summary: The player uses the settings menu to adjust the volume of the sound effects and/or music.

Actors: Player

Preconditions: Player has started the game and navigated to the settings menu.

Basic sequence:

Step 1: Player adjusts slider bars for SFX and/or Music.

Step 2: Player clicks "Save Settings" button.

Step 3: Selected settings are saved in settings file.

Exceptions:

Step 2: Player exits settings menu without clicking “Save settings” button: Settings remain unchanged.

Post conditions: SFX/Music volume is adjusted accordingly in-game.

Priority: 3.

ID: BIGGSSM01

Name: Dodge enemy projectiles and attacks

Summary: The player controls their character to avoid enemy projectiles and attacks.

Actors: Player

Preconditions: Player has started the game and is in a level with enemies.

Basic sequence:

Step 1: Player moves within an enemy’s line-of-sight.

Step 2: Enemy moves towards player until they’re within attack range.

Step 3: Enemy attacks towards player’s current position.

Step 4: Player dodges the attack.

Step 5: Go to **Step 2**.

Exceptions:

Step 2: Player moves out of enemy’s line-of-sight: Enemy returns to idle status.

Step 4: Player is hit by the attack: Player is affected by the enemy attack (e.g. takes damage, is applied a status effect, etc).

Any Step: Player defeats enemy: Enemy’s action is interrupted, and they perform their death routine.

Any Step: Player is defeated: Enemy’s action is interrupted, and they return to idle status.

Post conditions: Player is defeated, enemy is defeated, or player moves out of enemy’s line-of-sight.

Priority: 2.

ID: BIGGSPF01

Name: Use a heat-seeking weapon.

Summary: The player uses a heat-seeking weapon to target and fire at an enemy.

Actors: Player

Preconditions: Player has started the game and is in a level with enemies, with a heat-seeking weapon equipped.

Basic sequence:

Step 1: Player uses weapon to fire a projectile in a certain direction.

Step 2: Projectile travels in its assigned pattern (e.g straight-line/wiggly, fast/slow, etc.) until an enemy enters its attack range.

Step 3: Projectile turns towards closest enemy, adhering to its turning radius.

Step 4: Projectile hits enemy.

Exceptions:

Step 2-4: Projectile timer expires: Projectile explodes.

Step 3: Enemy stops being in projectile's attack range: Return to **Step 2.**

Step 4: Projectile fails to hit enemy: Return to **Step 3**.

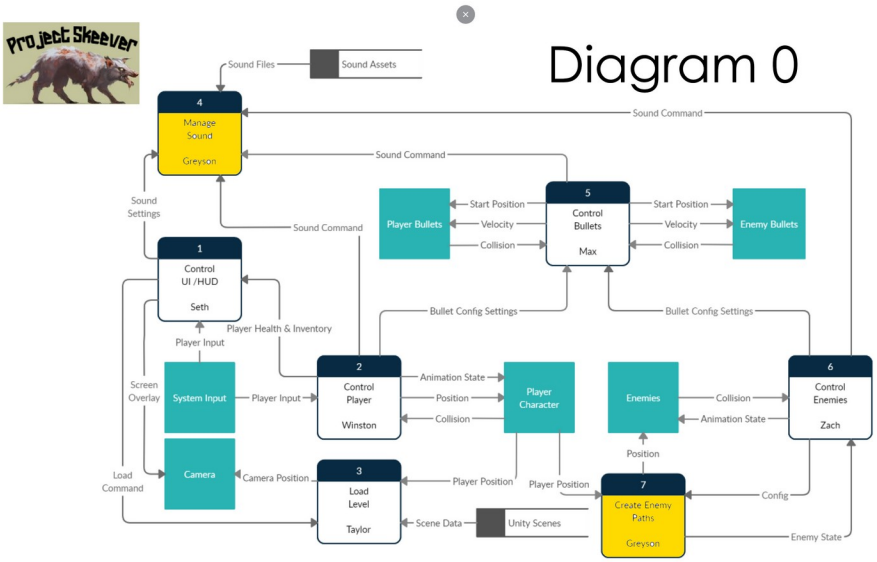
Post conditions: Projectile explodes.

Priority: 2.

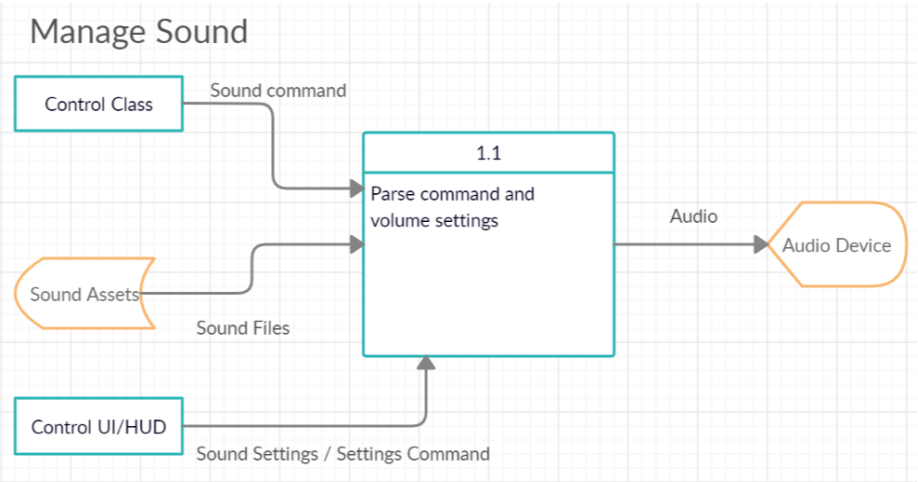
ID: BIGGSPF02

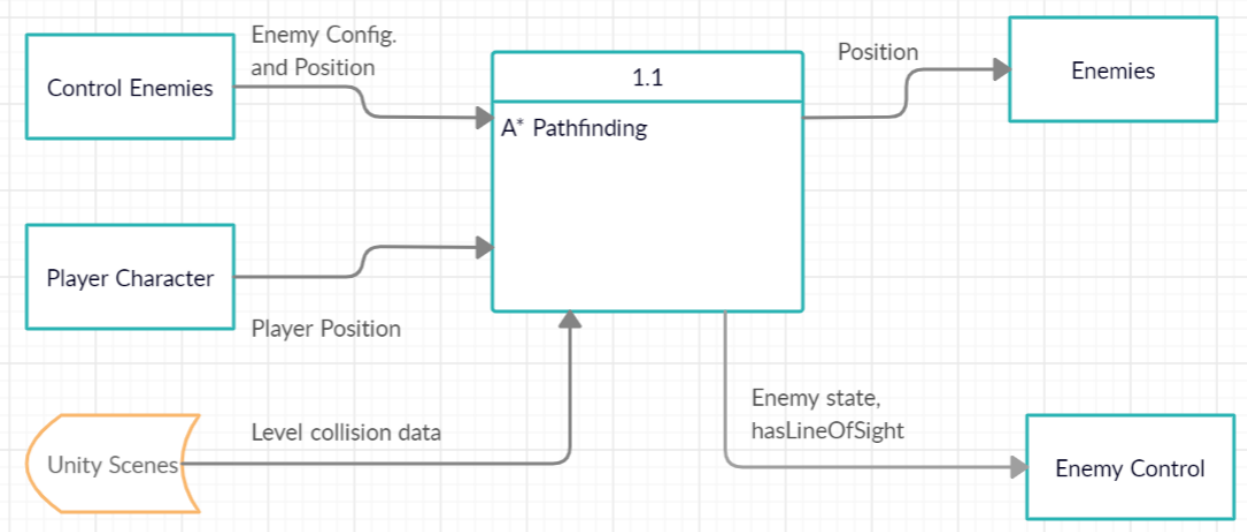
3. Data Flow diagram(s) from Level 0 to process description for your feature _____14

Data Flow Diagrams



(Note: Unity Scenes should have level collision data flowing to Create Paths)





Process Descriptions

Parse command and volume settings:

Load sound assets and corresponding IDs into memory
Load Sound Settings
WHILE TRUE:
IF sound command:
IF sound command is SFX:
Play sound effect (pushing audio to audio device).
ELSE IF sound command is music:
Stop current music, then begin new music.
ELSE IF settings command:
Reload Sound Settings
END WHILE

Astar Pathfinding*:

Init():
Load enemy config and position, player position, level collision data.
Update():
Reload player position, enemy position
Calculate path from enemy to player
FOR enemy:
IF needsLineOfSight:
IF path.straight-line == TRUE:
hasLineOfSight = TRUE
ELSE hasLineOfSight = FALSE
ELSE hasLineOfSight = TRUE
enemy.state = direction given by path found
Update enemy.position

//Note: Projectiles can be classified as either players or enemies for the purposes of this algorithm, and will seek their opposite out.

4. Acceptance Tests _____9

Pathfinding:

Run test 100 times sending output to a file:
With the player character spawned in a random (legal) position on the map, have each enemy with line-of-sight move to the player's position.
Output file will have the following characteristics:

For each enemy with line-of-sight, success or failure.
Failure for a given enemy type must not occur more than 3 times.
Failure for all enemies must not occur more than 8 times.

Run test 5 times, with output hand-written:
With the player character spawned in a random (legal) position on the map, note each enemy that should have line-of-sight, and which ones actually do. There should be no more than 2 discrepancies.

Sound Manager:

Repeat this test 3 times:
Play the game in the following order: Menu → Level 1 → Level 2 → Level 3 → Boss → Ending → Menu. In addition, enter the pause menu in each level. Then, test going from the pause menu to the main menu. Music must properly transition between each scene and menu. May use debug mode.

Repeat this test 3 times:
Note for each enemy type whether, for each action that enemy could take, the correct sound effect triggered and played. May use debug mode. There should be no failures.

5. Timeline _____/10

The following is the development time table, Gantt chart, and Pert chart for Fiasco development as a whole. I am entirely in charge of Sound Design and will be partially responsible for Combat Implementation as well:

Task no.	Task	Start	Duration (hours)	Predecessor Task(s)	Slack Time (hours)
1	Requirements Collection	0	8		0
2	Scene Design	8	8	1	12
3	Character Design	8	20	1	0
4	Environment Design	8	20	1	0
5	Sound Design	28	8	1, 2, 3	27
6	UI/HUD Design	28	15	1, 3	20
7	Combat Implementation	28	35	3	0
8	Testing	63	30	4, 5, 6, 7	0
9	Platforming	93	10	8	0