Evan M Jones
Beeston
CS 383
September 19, 2019

Champion Document

1. Brief Introduction

I'll be implementing two features: (1) a particle system and (2) sound effects.

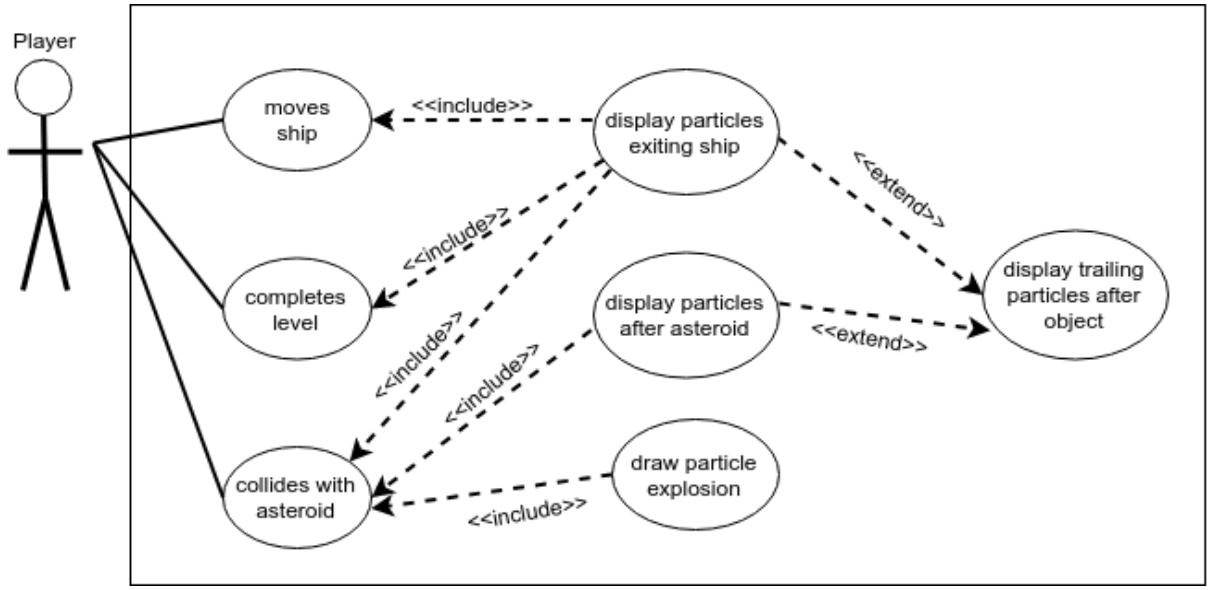
a. Particles

A particle system will be implemented to simulate a kind of "fuzzy" phenomena to certain elements found in game. Elements such as the user controlled spaceship, any stars in the level, and collisions between objects. This will be a three dimensional particle system meant to make movements and interactions of objects closer to real life. Particles themselves while seen will continue to be emitted and quickly fade out, as in a "fire" effects. The particles will start at a "base" and move outward -- signifying to users the direction of the given object.

b. Sounds

Various sounds will be implemented in game. Sounds such as: ambient menu/user interface soundtracks, songs to accompany a user through a level, collisions of the user with asteroids and other debris from the levels, and level begin/end sound clips. Most sounds will be triggered on specific user actions. When the user reaches the main menu user interface (for selecting difficulty, level, and settings) the ambient soundtrack will be triggered. When leaving the main menu it will stop. When a level begins the level's track/song will begin to play. When the level is completed or failed it will stop. Collision will cause an explosion sound to be made.

2. Use Case Diagram with Scenario



a. Particles

i. **Name:** Moves ship.
Summary: Particles will move a certain direction and velocity based on the user's trajectory.
Actors: The player.
Preconditions: A level must have been commenced.
Basic Sequence:

Step 1: The player will move move the controls, either moving up, down, left, right, or a combination of two.
Step 2: Particles will begin to be generated at the tail end of the user's ship.
Step 3: Particles will move out and away from the ship while fading away.

Exceptions:

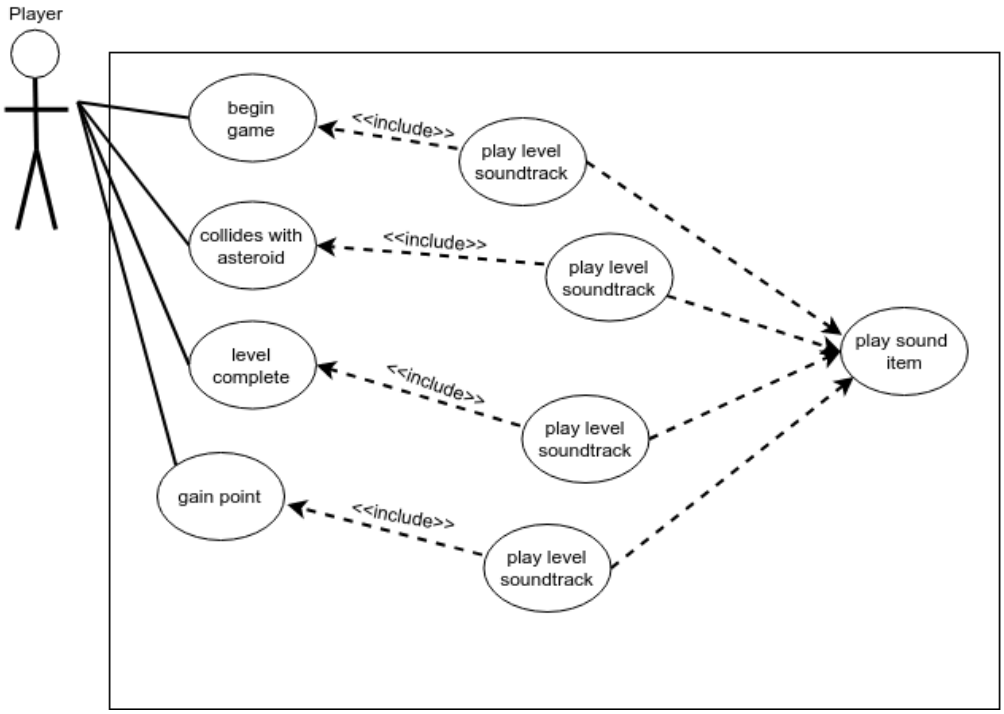
Step 1: End of game sequence has begun. No more user input is being taken.
Step 2: While the level credits are displayed to user, ship particles will still be drawn as the (now user uncrolledable) ship will be thrust forward through credits.

Post Conditions: Particles are seen as if exiting from the ship's thrusters.
Priority: 1
ID: P01

ii. **Name:** User ship collides with asteroid
Summary: An explosion of particles will occur when the user collides with an asteroid on the playing field.
Actors: The player.
Preconditions: A level must have been commenced.
Basic Sequence:
Step 1: The user's trajectory and an asteroid's trajectory meet.
Step 2: Both the asteroid and the user's ship will have their models replaced with their broken and destroyed counterparts.
Step 3: Many particles are drawn in an outward moving spherical shape from the point of impact.
Exceptions:
Step 1: End of game sequence has begun. No more user input is being taken.
Step 2: While the level credits are displayed to user, ship particles will still be drawn as the (now user uncrolledable) ship will be thrust forward through credits. No asteroid will be encountered after.
Post Conditions: Particles are seen "exploding" from the point of impact between the user's ship and the asteroid.
Priority: 1
ID: P02

iii. **Name:** User completed level.
Summary: The level has been completed and user impact is no longer used to move the ship.
Actors: The player.
Preconditions: The player must have successfully completed the level.
Basic Sequence:
Step 1: The user completes the level.
Step 2: Particles are seen exiting the ship's thrusters in a forward moving motion.
Step 3: The ship and particles are unresponsive to user controls.
Exceptions:
Step 1: Once end of game sequence has stopped the level will be, (1) reset or (2) exit'd.
Step 2: Particles will behave as the did in P01 or none will be observed at all (in the game's main menu).
Post Conditions: The game is over. The user must make a choice to continue to the next level or return to the main menu.
Priority: 2
ID: P03

b. Sound



i. **Name:** Level soundtrack
Summary: The song played during a level.
Actors: The player
Preconditions: The player is in game, and has navigated through the main menu to play.
Basic Sequence:
Step 1: Player enters game
Step 2: Chooses level/difficulting/any other settings.
Step 3: Level has started
Step 4: Level soundtrack begins.
Step 5: Level is either completed or lost. Soundtrack stops.
Exceptions:
Step 1: Player leaves mid level and returns to the main menu.
Step 2: Soundtrack stops.
Post Conditions: After song begins level will play with song until the song completes (game is over) or other conditions such as: player loses level or exits level. Soundtrack will not be playing afterwards.
Priority: 1
ID: S01

ii. **Name:** Collision with asteroid.
Summary: An explosion sound is heard if the player collides with an asteroid.
Actors: The player.
Preconditions: The player must be playing a level.
Basic Sequence:
Step 1: The player moves themselves into the trajectory of an asteroid.
Step 2: The asteroid makes contact with the player's ship.
Step 3: Explosion sound is player.
Exceptions:
Step 1: Player pauses or leaves the game mid collision with asteroid.
Step 2: Explosion sound is paused during pause menu, starts to play from where it left off after unpause -or- explosion sound is stopped completely if the game is exited.
Post Conditions: The player heard the explosion sound and is aware they collided with an asteroid. The player has now lost the level.
Priority: 1
ID: S02

iii. **Name:** Level complete jingle.
Summary: A noticeable jingle is heard by the player when the level has been completed.
Actors: The player.
Preconditions: The player successfully completes a level.
Basic Sequence:
Step 1: The player survives and completes a level.
Step 2: The jingle is immediately played.
Exceptions:
Step 1: The player pauses or exits the game mid jingle.
Step 2: If the game is paused the jingle will resume when the player unpauses. If the game is exited the jingle will completely stop.
Post Conditions: The player hears the jingle and is aware the level has been completed. The player is happy.
Priority: 2
ID: S03

iv. **Name:** Point gained jingle.
Summary: The player has gained a point and a jingle is played.
Actors: The player.
Preconditions: The player must score a point.

Basic Sequence:

Step 1: The player interacts with a “ring” like structure to score a point. Travelling through the ring.

Step 2: Immediately when the user travels through the ring a jingle is heard.

Exceptions:

Step 1: The player pauses or exits the game mid jingle.

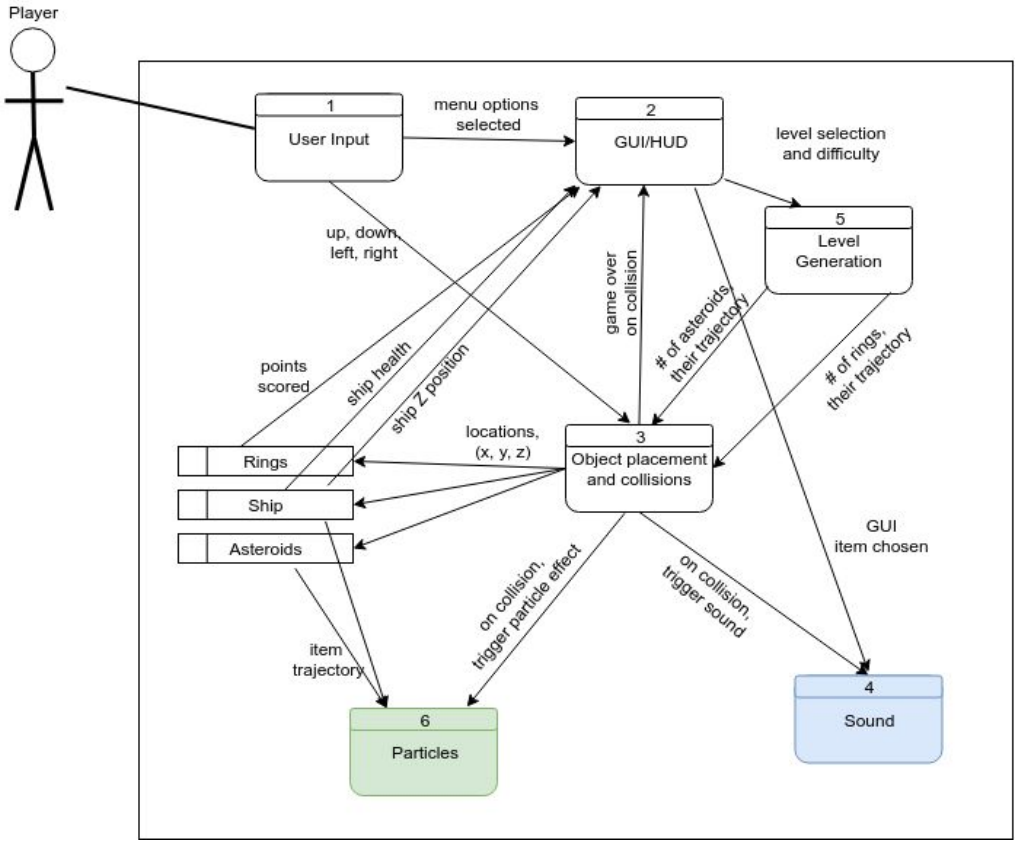
Step 2: If the game is paused the jingle will resume when the player unpauses. If the game is exited the jingle will completely stop.

Post Conditions: The player has heard the jingle and is aware they scored a point. The player is happy.

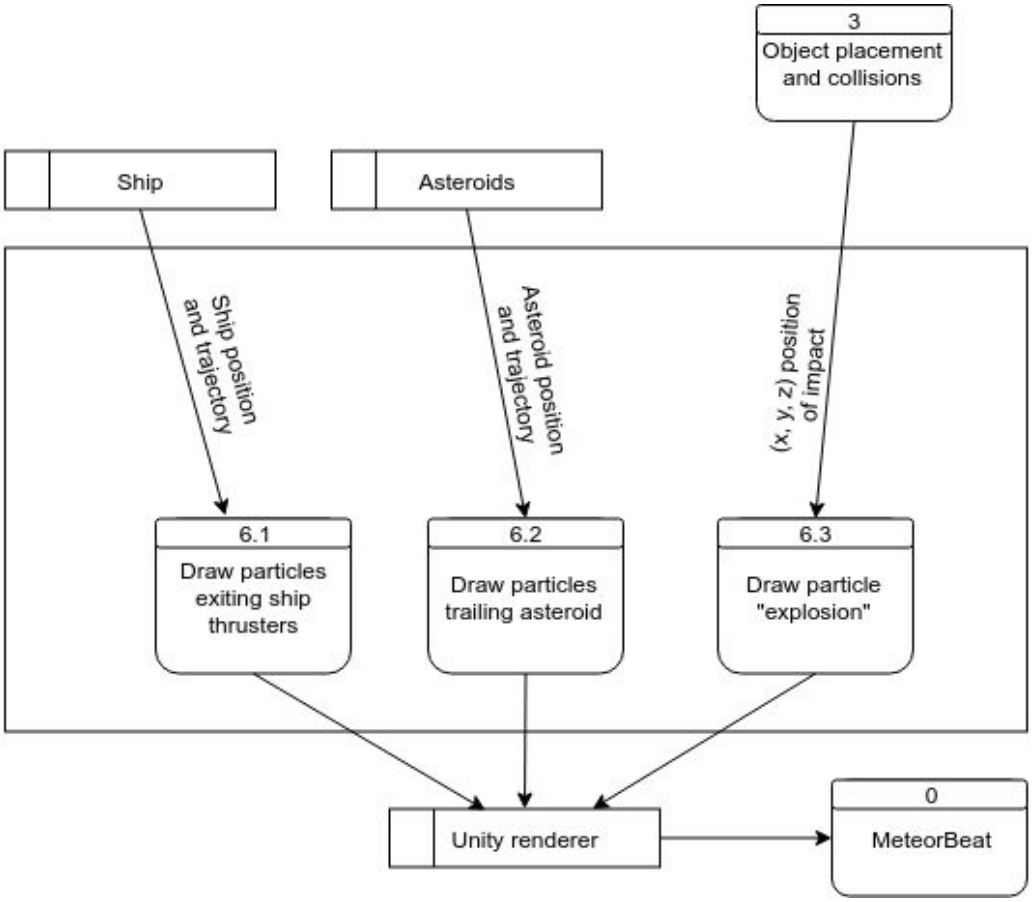
Priority: 2

ID: S04

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



Particles:



Process descriptions:

6.1 Draw particles exiting ship thrusters:

IF ships is moving at a velocity greater than zero

Draw particles exiting ship.

Calculate spread in ship particles.

Draw some particles exiting straight out the back.

Draw other particles exiting with a 1 to 3 degree slant.

Rotate particles trajectory by random amount:

Math.random() * 360 around velocity vector of ship.

As particles move away from ship fade out particles

When particles have been completely faded remove from scene

WHILE time (time from current particle spread) < 500 milliseconds

Continually push the particles generated in the opposite

Direction of the ship.

Opacity is decreased by 0.01 unit.

ENDWHILE

ELSEIF

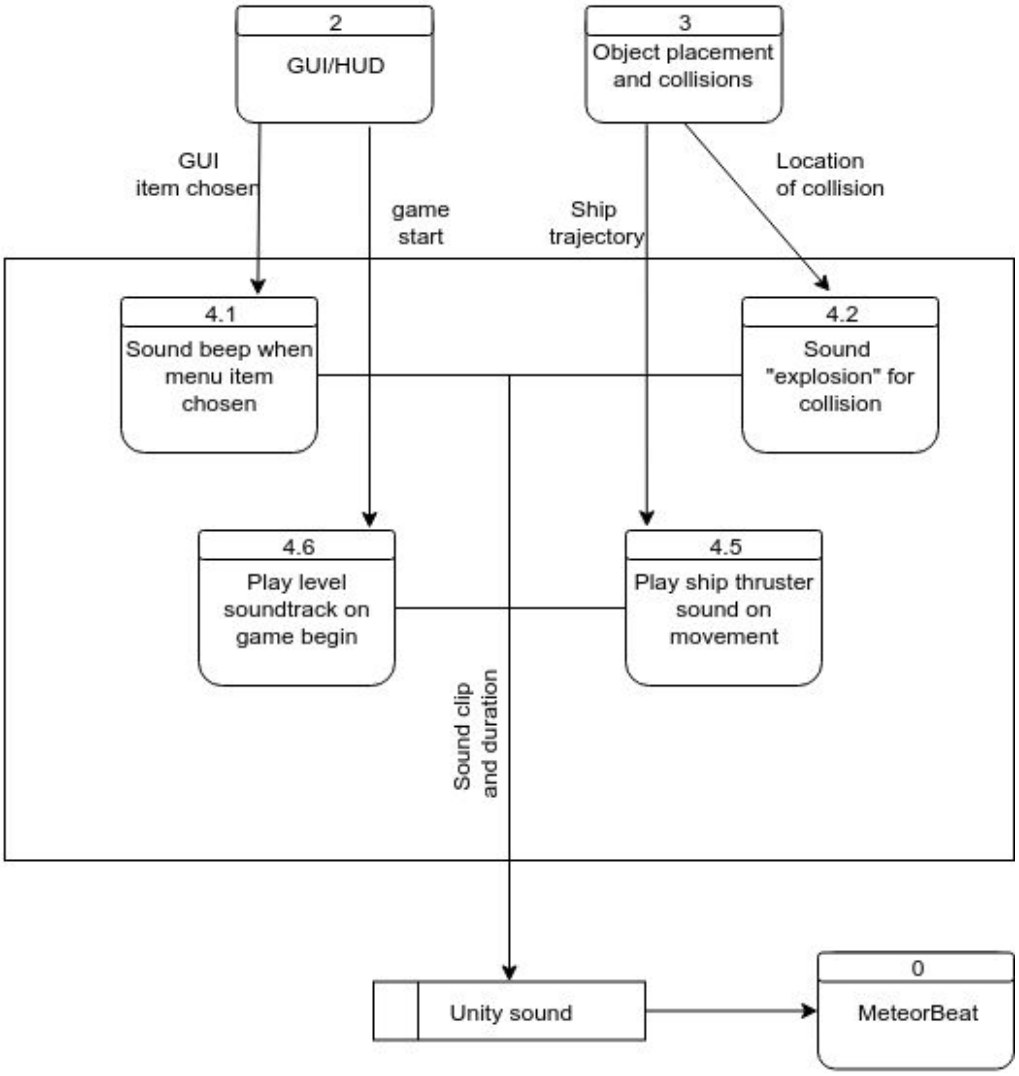
Stop drawing particles exiting ship thrusters

```
        Fade out all existing particles near ship
    ENDIF

6.2 Draw particles trailing asteroid
    WHILE asteroid is in player view
        Draw particle stream trailing asteroid movement.
            Calculate spread in asteroid particles.
            Draw some particles exiting straight out the back.
            Draw other particles exiting with a 1 to 3 degree slant.
            Rotate particles trajectory by random amount:
                Math.random() * 360 around velocity vector of asteroid.
        As particles move away from asteroid fade out particles
            When particles have been completely faded remove from scene
        WHILE time (time from current particle spread) < 500 milliseconds
            Continually push the particles generated in the opposite
            Direction of the asteroid.
            Opacity is decreased by 0.01 unit.
        ENDWHILE
    ENDWHILE

6.3 Draw particle “explosion”
    IF object 1 == ship AND object 2 == asteroid
        Draw particle “sphere” from collision point.
            Calculate spread of particles.
            LOOP ParticleCount = Math.random() * 1000
                Calculate trajectory of current particle from collision point.
                X = Math.random() * 360
                Y = Math.random() * 360
                Z = Math.random() * 360
                Speed = 500 units
                Particle = CreateParticleObject()
                Particle.position = collision point
                Particle.velocityVector = (X, Y, Z)
            ENDLOOP
        WHILE time (time from current particle spread) < 500 milliseconds
            Opacity is decreased by 0.01 unit.
        ENDWHILE
    ENDIF
```

Sound:



Process descriptions:

4.1 Play sound beep when menu item is chosen.

```
FOR clickedItem
    IF hasAssociatedSound(clickedItem)
        findAssociatedSound(clickedItem).play()
    ELSEIF
        playDefaultMenuItemClickedSound()
    ENDIF
ENDFOR
```

4.2 Play sound “explosion” for collision.

```
UNTIL GameIsOver OR time > 1 second
    /* player can quick quick the game from this point, or restart */
    IF object 1 == ship AND object 2 == asteroid
        X = explosionSounds
        X[Math.floor(Math.random() * (X.length - 1)].play()
    ENDIF
ENDUNTIL
```

4.3 Play level soundtrack on game begin.

```
Song = PossibleSongs[Math.floor(Math.random() * PossibleSongs.length - 1)]
Song.play()
WHILE NOT GameIsOver()
    /* Loop until game is over and end sound */
    Song.stop()
ENDWHILE
```

4.4 Play ship thruster sound on movement.

```
IF player.velocity() > 0
    Set = possibleThrusterSounds
    RandomThrusterSound = Set[Math.floor(Math.random() * Set.length - 1)]
    player.thruster.playSound(RandomThrusterSound)
ELSEIF
    player.thruster.stopSound()
ENDIF
```

4. Acceptance tests

a. Particles
   i. Ensure particles move correct direction
   Input: object, velocity vector, position
   Output: Stream of particles
   Notes: Resulting stream of particles must be moving opposite of velocity vector at the given position.

Output	X vector	Y vector	Z vector	Notes
Particles stream from (0,0,0) to (-1,-1,-1)	1	1	1	Should draw particles from (0, 0, 0) to (-1, -1, -1) as if exiting ship.
Particles stream from (0,0,0) to (-1,-1,-1)	-1	1	1	Should draw particles from (0, 0, 0) to (1, -1, -1) as if exiting ship.
Particles stream from (0,0,0) to (1,1,-1)	-1	-1	1	Should draw particles from (0, 0, 0) to (1, 1, -1) as if exiting ship.
Particles stream from (0,0,0) to (1,1,1)	-1	-1	-1	Should draw particles from (0, 0, 0) to (1, 1, 1) as if exiting ship.
Particles stream from (0,0,0) to (-1,-1,1)	1	1	-1	Should draw particles from (0, 0, 0) to (-1, -1, 1) as if exiting ship.
Particles stream from (0,0,0) to (1,-1,1)	-1	1	-1	Should draw particles from (0, 0, 0) to (1, -1, 1) as if exiting ship.

ii. Object collision particles
Input: Two objects.
Output: Explosion of particles
Notes: An abundance of particles will be created at the point of impact between two objects.

Output	Point of impact	Radius	Notes
A growing and then	(x, y, z), let	r, let r be a	An explosion should

fading out “explosion”/sphere of particles should be placed at (x, y, z), starts fading as the sphere approaches the radius, r.	x,y,z all be float values.	float value.	be viewable from the point of impact and it will grow until the sphere of particles reaches the radius given, then should fade out.

Note: exact values were not specified here as it would be best to “fuzz” test this and supply random values each time.

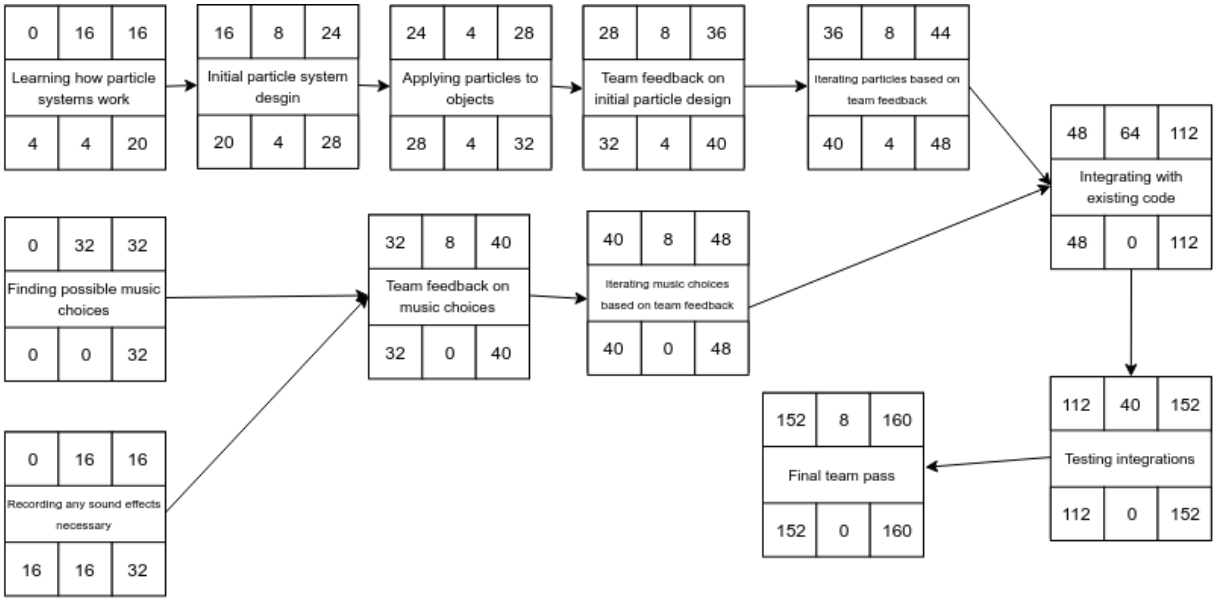
b. Sounds
    i. Ensure sound plays at the correct time, no noticeable delay or lag. Stops appropriately.
    Input: Sound clip, moment to start, moment to end.
    Output: Smooth playing sound, starts and ends at the given times
    Notes: There will be no noticeable loss of quality, delay, or log associated with the sound clip being played.

Output	Sound clip	Duration	Notes
The “mocked” platform API should report that the sound did play and for the correct duration, fail otherwise.	A sound clip from the assets.	A float, t.	This test should not actually play a sound, but should be passed a sound “provider”/mock that would mock the platform APIs to play the song. In a test environment it wouldn’t be appropriate to play sounds on a real device. Repeat this test for all sound files in assets.

5. Timeline

Task #	Task name	Time estimation	Depends on
1	Learning how particles systems work	16	
2	Initial particle design	8	1
3	Applying particles to game objects	4	2
4	Team feedback on initial design	8	3
5	Iterating particles system based on team feedback	8	4
6	Integrating with existing code	64	5, 11
7	Testing integrations	40	6
8	Finding possible music choices	32	
9	Recording any sound effects necessary	16	
10	Team feedback on initial music choices	8	9
11	Iterating music choices based on team feedback	8	10
12	Final team pass	8	7

PERT


0 16 16
Learning how particle systems work
4 4 20
16 8 24
Initial particle system desgin
20 4 28
24 4 28
Applying particles to objects
28 4 32
28 8 36
Team feedback on initial particle design
32 4 40
36 8 44
Iterating particles based on team feedback
40 4 48
48 64 112
Integrating with existing code
48 0 112
0 32 32
Finding possible music choices
0 0 32
32 8 40
Team feedback on music choices
32 0 40
40 8 48
Iterating music choices based on team feedback
40 0 48
0 16 16
Recording any sound effects necessary
16 16 32
152 8 160
Final team pass
152 0 160
112 40 152
Testing integrations
112 0 152

GANTT


Task Name
Hours
0 4 8 12 16 20 24 28 32 36 40 44 48 52 56 60 64 68 72 76 80 84 88 92 96 100 104 108 112 116 120 124 128 132 136 140 144 148 152 156 160
Complete project execution
Particle related
Learning how particle systems work
Initial particle system design
Applying particles to objects
Team feedback on initial particle design
Iterating particles based on team feedback
Sound related
Find possible music choices
Recording any sound effects necessary
Team feedback on initial music choices
Iterating music choices based on team feedback
Shared
Integrating with existing code
Testing integration
Final team pass