Asteroids Technical

Dawid Jerdonek

Davids Jalisevs

Adam Cashin

Donal Howe

Hardware + Software used:

School Computers

Visual Studio

SFML

Photoshop

Audacity

Asset Pipeline

1. Decide what assets we need
2. Begin initial sketches and designs
3. Make the final sketches
4. Begin work on Photoshop
5. Edit the Photoshop concepts
6. Produce final renders with transparent backgrounds for assets
7. Implement the assets into the game
8. Make any needed changes such as size or positioning

Code Development

1. Create Game States for Menu screen, How to Play screen and Gameplay. Done the same way Mario Cart game states.
2. Connect the Game States
3. Create classes for the game to be used for gameplay.
4. Begin work on the gameplay, add a simple sprite and make it move around the screen using arrow keys and WASD keys.
5. Add 2 different enemy types an asteroid and an enemy
6. Make the enemy type Asteroid
7. Make the Asteroid spawn at a random position on top of the screen and draw a vector towards a random position at the bottom of the screen.
8. Give the Asteroid a Velocity and make it move along its vector.
9. Code the other enemy so that it follow the player and shoots in the players direction.
10. Have the enemy shoot using code similar Lab 3. The shot moves along a vector from enemy position to player position at time of shot.
11. Implement the assets which we have created and apply them to the gameplay
12. Create collisions between asteroids and the player and the enemy shot and the player
13. Create a shoot function for the player. Similar to enemy shot but the end position will be based on the direction which the player is facing.
14. Create collisions between player shot and enemy and between player shot and asteroids and between player and enemy.
15. Create a scoring system.
16. Add Instructions to the How to Play screen.
17. Add all necessary options to Main Menu.

18. Add a Galaxy Map game state.
19. Add images and assets to the map
20. Connect the map to the gameplay
21. Add levels based on simple integers. For example if the integer is 2 the enemy count and speed is increased by 2.
22. Add The remaining Game States and connect them to the existing ones.
23. Create and upgrades system also based on integers, for example if fire rate is upgraded to 3, the speed of the player shot is increased by 3.
24. Complete the basic gameplay functionalities. Make the player rotate, make his velocity go down gradually rather than making him stop instantly and add a currency system.
25. Add sound files to the game
26. Work on the boss battle.
27. Draw the boss sprite and make him move randomly around the screen.
28. Make the enemy shoot outward in multiple directions so that the player must dodge the shots
29. Implement collisions between the enemy shots and the player and between the enemy sprite and the player.
30. Implement collisions between boss and player shot.
31. Create a health variable for the boss and decrement by 1 each time he is shot.
32. The player will complete the game if the boss is defeated so program a Congratulations screen.
33. Add any extra functionalities such as Game Saves, different ships, High scores, More upgrades, and an endless mode.

Plan and Schedule

Create Game States and Begin work on Gameplay – March 13th

Add and connect all Game States – March 20th

Finish Gameplay Prototype – March 30th

Basic Gameplay functionality edits – April 3nd

Add all graphics, assets and sounds – April 10th

Finish Upgrade system and basics of all Game States - April 17th

Complete galaxy Map and boss battle - April 24th

Some work on extra functionalities - May 1st

Complete Game with all extra functionalities - May 3rd

Class diagram

Player Class	Player Bullet	Boss Bullet
sf::sprite m_playerSprite; sf::texture m_playerTexture; int m_playerHealth float m_playerVelocity bool m_playerMoves = false; functions void draw; void update; void move; void boundaryCheck; void setBody:	sf::sprite playerBulSprite; sf::texture playerBulTexture; float m_speedOfBullet; bool m_bulletShot; bool m_bulletMoving; functions void drawBullet; void setBullet; void bulletFired; void bulletBoundry; void bulletCollisions;	sf::sprite bossBulletSprite; sf::texture bossBulletTexture; float m_bossSpeedBul; bool m_bossBulShot; bool m_bossBulMoves; bool m_bulletBossShow; int bulletTime; sf::soundBuffer m_bossShotBuffer; sf::sound m_bossShotSound m_shootSound; functions void update; void draw; void bulletFired; void bulletCollisions; void billetboundary;

asteroidEnemy class	alienEnemy class	Boss class
sf::sprite asteroidSprite;	sf::sprite alienSprite;	sf::sprite bossSprite;
sf::texture asteroidTexture;	sf::texture alienTexture;	sf::texture bossTexture;
float m_asteroidSpeed;	float m_alienSpeed;	float m_bossSpeed;
sf::vector2f m_heading;	sf::vector2f m_heading;	sf::vector2f m_heading;
sf::vector2f m_velocity;	sf::vector2f m_velocity;	sf::vector2f m_velocity;
sf::vector2f m_startPoint;	sf::vector2f m_startPoint;	sf::vector2f m_startPoint;
sf::vector2f m_endPoints;	sf::vector2f m_endPoints;	sf::vector2f m_endPoints;
functions	functions	functions
void draw;	void draw;	void draw;
void movement;	void movement;	void movement;
void update;	void update;	void update;
void setAsteroid;	void setAlien;	

upgrade Class;	Game Class;	
sf::sprite upgradeOneSprite sf::sprite upgradeTwoSprite sf::sprite upgradeThreeSprite sf::sprite upgradeFourSprite (approx. might do in the ARRAYS); sf::texture upgradeOneTexture sf::texture upgradeTwoTexture sf::texture upgradeThreeTexture sf::texture upgradeFourTexture int upgradeStats[UPGRADES] const int UPGRADES (4) // for new funstions void incrementUpgrades void decrementUpgrades	Run; Int main; Player myPlayer; PlayerBullet playerBul; BossBullet bossBul; AlienEnemy alienShip; Asteroids asteroidEnemy; int processEvents;	