Assignment 3 - Asteroids!

Authors

Matt Watts, Christopher Davidson, Marcus Girard

Design philosophy

We designed the procedural implementation of our game to behave similarly to the Atari 2600 version of the game. We designed the object oriented implementation of our game to behave similarly to the version in the youtube video on the assignments moodle page.

We originally wrote the game using a procedural design methodology. We then rewrote the game from scratch using an object oriented design methodology to better facilitate the expanding scope of the project. This document mostly describes the procedural implementation of the game.

We spent a lot of time testing at each stage of the implementations to ensure the software was robust and error free.

User interface

The graphical display of the game has a 3x4 aspect ratio which is the same as old school arcade consoles.

At the start of the game, your ship appears stationary at the centre of the display pointing up.

Four asteroids appear on-screen. The asteroids move at a fixed speed. They appear randomly in the left and right quadrants of the display moving close to an up or down direction. This ensures the ship won't be hit by an asteroid during the first part of the game. When an asteroid reaches the edge of the display, it wraps around so it appears diagonally opposite on the corresponding edge of the display.

The current score appears in the top left corner of the display.

The current level appears in the display under the score.

The current number of spare lives remaining appear under the level as icons.

Controls

The keyboard is used to control game-play.

Key	Usage
Up arrow	Accelerate ship
Down arrow	Decelerate ship
Left arrow	Rotate ship left
Right arrow	Rotate ship right
Space bar	Fire laser bolt
Enter	Teleport ship

Ship movement

When the user uses the arrow keys to rotate and accelerate the ship, the ship moves with realistic physics around the display. When the ship reaches the edge of the display, it wraps around so it appears diagonally opposite on the corresponding edge of the display.

Ship exhaust

When the ship accelerates, a visible exhaust appears behind the ship. When the ship decelerates, a visible exhaust appears in front of the ship.

Ship lives

Icons representing the remaining lives appear in the display under the level. The user has four lives at the start of the game, so three icons appear to indicate the three spare lives. When a ship is destroyed, the number of lives decreases by one and fewer icons are displayed.

On a new life, the ship returns to the centre of the display and is stationary.

The ship doesn't re-spawn until the area around the centre of the display is free from aliens. If the wait is too long, the ship spawns anyway.

If there are no remaining lives, the game ends.

Alien visits

In level 2 and above, an alien randomly spawns. It approaches from either the left or right of the display randomly, then moves at a fixed speed across the display until it disappears off the side of the display. The alien ship randomly fires a laser bolt during it’s visit.

If the alien laser bolt hits the ship, the ship is destroyed. If a ship laser bolt hits the alien, the alien is destroyed. If the alien collides with the ship, the ship is destroyed.

Scoring

When the user presses the Space bar, the ship fires a laser bolt that moves at a fixed speed in the direction the ship was facing when it fired the bolt.

If a laser bolt hits an asteroid, the asteroid is destroyed and the score increases by an amount that depends on the size of the ship hit. You get more points for hitting a smaller asteroid.

If a laser bolt hits an alien, the alien is destroyed and the score increases by 200 points.

The laser bolt travels for a number of pixels equal to the display height. Many laser bolts can be in flight simultaneously. There's a slight pause after a laser bolt fires before the ship can fire another laser bolt.

Object Shot	**Score**
Large Asteroid	25
Medium Asteroid	50
Small Asteroid	100
Alien	200

Teleporting

When the user releases the Enter key, the ship teleports. It disappears and reappears at a random location on the display. The ship is stationary when it reappears. Any motion the ship had before teleporting is lost.

Explosions

When the ship collides with an asteroid, collides with an alien, or is shot with an alien laser bolt, it is destroyed and an explosion animation plays at the location of the ship.

When a ship laser bolt collides with an asteroid, the asteroid is destroyed and an explosion animation plays at the location of the destroyed asteroid.

When a ship laser bolt collides with an alien, the alien is destroyed and an explosion animation plays at the location of the destroyed alien.

Spawning

When a large asteroid is destroyed, two medium asteroids are spawned in its place. Medium asteroids travel faster than large asteroids.

When a medium asteroid is destroyed, two small asteroids are spawned in its place. Small asteroids travel faster than medium asteroids.

When a small asteroid is destroyed, no asteroids are spawned.

The location of a spawned asteroid is the same as the asteroid destroyed, and the direction of a spawned asteroid is random.

Level up

When all of the asteroids are destroyed, the game goes to the next level.

The ship returns to the centre of the screen and is stationary.

The asteroids re-spawn the same as at the start of the game, except that the asteroids move faster than they did on the last level so the difficulty of the game increases.

More asteroids appear on each level up.

Aliens move faster on each level up.

Game over

When all the ships lives are destroyed, the game is over.

A game over message displays and the user can press any key to restart the game.

If the user has a key pressed down when the game ends, a small pause causes the game not to start straight away so the user can see the game over message.

Game parameters

There's some game parameters that can be set in the Processing source code that affect game play.

Parameter	Value	Affect
laserSpeed	30	determines how fast the laser bolt moves
thrustConstant	0.2	pixels per frame per frame ship acceleration
asteroidStartSpeed	1	how fast large asteroids move on level 1
asteroidSpeedChange	0.25	how fast asteroid speed increases with difficulty increase
alienSpeedChange	0.5	how fast alien ship speed increases with difficulty increase
alienLaserSpeed	5	speed of alien laser bolt
lives	4	how many lives do you have
laserSize	10	size of a laser bolt
startAsteroidCount	4	how many asteroids does level 1 start with?
asteroidLevelUp	2	how many extra asteroids on new levels
shipSize	10	radius size on ship triangle
shipTurnSpeed	50	determines how quickly ship turns when left and right arrow pressed
centreRadius	100	detection radius for "is asteroid close to display centre"
laserFireFrames	5	how many frames must pass before laser fires again
endGameFrames	20	how many frames must pass before a new game can start
alienSpawnSeconds	10	stochastic delay before alien ship spawns
alienSize	25	size of alien ship
alienLevel	2	level that alien first appears on
alienLaserSpawnSeconds	10	stochastic delay before alien ship fires laser bolt
alienLaserSize	10	size of alien laser bolt
scoreAlien	200	points for shooting alien ship
scores	{25,50,100}	points for shooting {large, medium, small} asteroids
asteroidSizes	{75,50,25}	diameter for each asteroid circle: {large, medium, small}
spawnWaitLimit	300	how many frames must pass before a new ship is forced to spawn
laserFireFrames	5	how many frames must pass before laser fires again
endGameFrames	20	how many frames must pass before a new game can start
frameRate	30	frames per second
initialShipDirection	1.5*PI	initial ship direction is up: 1.5 pi radians is up

There’s also some parameters that specify the explosion animations.

Parameter	Value	Affect
shipExplosions	{500,450,400,350,300,250,200,150,100,50}	ship explosion "frames"
asteroidExplosion	{250,225,200,175,150,125,100,75,50,25}	asteroid explosion "frames"
alienExplosions	{250,225,200,175,150,125,100,75,50,25}	alien explosion "frames"

Each element of each array specifies an explosion "frame" diameter. A frame is a circle of specified diameter. Explosion frames can be added or removed by simply adding or removing elements to the fixed size arrays. The last element in the arrays is the first frame of the animation, and the first element in the arrays is the last frame of an animation. When one of these objects explodes, an explosion animation starts at the location the object exploded.

The display size is set to a width of 500 and a height of 650 pixels which is a 3x4 aspect ratio.

Game parameters

In addition to the game parameters, there are a number of variables used to represent the current game state.

Table of functions

The game has 48 functions divided into 12 groupings.

	Grouping	Count
1	initialise	5
2	draw	15
3	move	6
4	collision	8
5	wait	3
6	explode	3
7	keys	3
8	accelerate	1
9	life	1
10	spawn	1
11	hit	1
12	fire	1

Table: 12 function groupings.

I	J	Function	Grouping	Description
1	1	setup	initialise	initialise the game
2	2	initGame	initialise	initialise the PVectors
3	3	restartGame	initialise	start or restart the game
4	4	initAsteroids	initialise	create new asteroid set for the start of a level
5	5	initKeys	initialise	set all keys to up (not pressed)
6	1	draw	draw	draw a frame of the game
7	2	drawGameOver	draw	draw the game over message
8	3	drawScore	draw	draw the score
9	4	drawLives	draw	draw the number of lives remaining as ship icons
10	5	drawAsteroids	draw	draw asteroids
11	6	drawExhaust	draw	draw a rocket exhaust for the ship
12	7	drawShip	draw	draw the ship
13	8	drawShipExplosion	draw	draw frame of ship explosion animation
14	9	drawAsteroidExplosion	draw	draw frame of asteroid explosion animation
15	10	drawAlienExplosion	draw	draw frame of alien explosion animation
16	11	drawExplosion	draw	draw frame of an explosion animation
17	12	drawPolygon	draw	draw a polygon
18	13	drawLaser	draw	draw the ships laser bolts
19	14	drawAlienLaser	draw	draw alien laser bolt
20	15	drawAlien	draw	draw alien ship
21	1	moveAsteroids	move	move the asteroids
22	2	moveShip	move	move the ship
23	3	teleportShip	move	teleport ship to a random location
24	4	moveLaser	move	move the ships laser bolts
25	5	moveAlienLaser	move	move alien laser bolt
26	6	moveAlien	move	move alien ship
27	1	detectAsteroidCentre	collision	detect if a asteroid is "near" centre of display
28	2	detectShipCollisionAsteroid	collision	detect collision between ship and asteroids
29	3	detectShipCollisionAlien	collision	detect collision between ship and alien
30	4	detectShipCollision	collision	detect ship collisions
31	5	detectLaserCollisionAsteroid	collision	detect collision between ship laser and asteroids
32	6	detectLaserCollisionAlien	collision	detect collision between ship laser and alien
33	7	detectLaserCollisionShip	collision	detect collision between alien laser and ship
34	8	detectLaserCollision	collision	detect laser collisions
35	1	waitToSpawn	wait	wait to spawn a new life for the ship
36	2	waitAlienLaser	wait	randomly spawn an alien laser bolt
37	3	waitAlien	wait	randomly spawn alien ship if it doesn't exist
38	1	shipExplodes	explode	make a ship explosion
39	2	asteroidExplodes	explode	make an asteroid explode
40	3	alienExplodes	explode	make an alien explode
41	1	processKeyPress	keys	do key action for keys that are currently pressed
42	2	keyPressed	keys	detect when a key is pressed down
43	3	keyReleased	keys	detect when a key is released
44	1	accelerateShip	accelerate	accelerate or decelerate the ship
45	1	nextLife	life	the ship moves to one of its remaining spare lives
46	1	spawnAsteroid	spawn	spawn a baby asteroid
47	1	asteroidHit	hit	an asteroid has been hit with a laser bolt
48	1	fireLaser	fire	fire a laser bolt

Table: 48 functions.

How you implemented the various features

Data structures

- PVectors to represent the location and velocity of the ship, alien, alien laser, and explosions

- ArrayLists of PVectors to represent location and velocity of the ship laser bolts

- Floats, Integers, Booleans, and Arrays to represent game variables and parameters

- ArrayLists of PVectors, Booleans, Floats and Integer's to represent asteroid variables

Methods

- Initialise functions to initialise variables for the start of the game, new ship lives, and new levels

- Draw functions to draw elements of the game composition to the display for each frame of the game

- Move functions to animate movement of the ship, asteroids, laser bolts, and the alien

- Collision detection functions with rectangular and circular collision detection to detect collisions between the ship, asteroids, laser bolts, and the alien

- Wait functions to spawn new elements of the game composition during play, for the ship, the alien, and alien laser bolts

- Explode functions to start explosion animations for the ship, asteroids, and the alien

- Keys functions to detect and process key press and key release events so the user control of the game is responsive and fluid

- Accelerate function to accelerate and decelerate the ship when the user presses the up or down arrow key

- Life function to move the ship to one of it’s remaining lives

- Spawn function to spawn an asteroid

- Hit function to control behaviour when an asteroid is destroyed

- Fire function to fire the ships laser bolt

Documentation

- Libre Office to create the game documentation

Overview video

- Kazam to record video screen shots, and Shotcut to compose the overview video

Design decisions that lead to your implementation

We based the design of the procedural implementation of the video game on childhood memories of playing asteroids on the Atari 2600. The object oriented implementation was designed to be more similar to the version in the youtube video on the assignments moodle page.

We chose values for game parameters that would make the game fun, and to make it more difficult as game play proceeds and level ups occurred.

Outline which parts of the project were contributed by each team member

Matthew Watts

- designed the procedural implementation of the video game
- wrote the procedural implementation
- tested and debugged the procedural implementation
- wrote the documentation
- created the git repository
- facilitated the overview video
- created components of the overview video

Christopher Davidson

- designed the object oriented implementation of the video game
- wrote the object oriented implementation
- tested and debugged the object oriented implementation
- created the slack channel for team communication
- created components of the overview video

Marcus Girard

- wrote the level counter, lives counter and ship lives implementation
- extensively tested the implementations
- facilitated the design of the game-play
- lead the design decisions to replicate the Atari 2600 version of the game
- designated and assigned tasks to team members
- created the activity diagram

Overcame challenges faced through the project

Teamwork

We had problems of communication and task scheduling. Each of us had competing priorities including heavy uni workloads, work commitments, and family commitments.

Complexity

As the program became larger and more complex, it became increasingly difficult to ensure the software was running robustly. The program became much larger and it was more difficult to find variable and function references. The program became more complex and it was more difficult to find and identify programming errors.

Overcoming challenges

There were a range of techniques we used to overcome these challenges and to better facilitate team programming, including:

- A variety of communication methods: email, a slack channel, a git repository, regular meetings between team members

- Having a team leader to lead the design, and designate and assign tasks to team members

- Improved commenting and documentation

- Revised designs and naming of variables and functions

- Frequent iterations of programming, testing, and discussion

- Rewriting the program in an object oriented framework to better facilitate the expanding scope of the project