Audio Driven Games

Masters Thesis

Documentation

Ashley Morrison (i7861963@bournemouth.ac.uk)
Msc Computer Animation & Visual Effects
N.C.C.A Bournemouth University
18th August 2009

1) ObjectSpace

Author: Ashley Morrison
Class: Main/ObjectSpace
Brief: Main program file containing setup,loop and input functions.
Contains game, menu and audio objects and flags to move user between screens.

1. Variables

main - class containing text outputs for the start and game over screens.

game - class of the game object containing all game related methods/items.

newGUI - class that uses the ControlP5 library for loading in and displaying the relevant user interface. Includes main menu, in-game settings and in-game player stats panels.

titleSong - class containing all the related minim audio library objects and calls for playing and analysing the audio for the main menu and in-game song as well. This object is for the main menu.

laser - minim library class item for loading in and accessing the laser sound effect played when the user fires the weapon.

explosion - minim library class item for loading in and accessing the explosion sound effect played when the player collides with an asteroid.

NewBackground - class containing the Spiral Galaxy objects used for the main menu visualisation.
R - default radius used for the sphere around which most items are distributed.

NoStars, **NoGalaxies** - Default values displayed for the menu gui for how many stars and galaxies will be drawn in game.

wormholeNoStars - Default value displayed in the menu gui for how many points to be used to draw the wormhole object.

wormholeRmax - Default value for the radius of the wormhole used in the render method for distributing the points.

galaxyParticleNo - Default value for the number of lines used to render the Pulser objects to screen.

galaxyRadius - Default value for the radius of the Pulser object particles to be distributed around.

beatSensitivity - Default value for the beat detection dampening amount. Larger value means more dampening, less beats detected, less asteroids generated.

filename - Empty string value for the filename of the song to use for the game. If left empty, a default song is set in the game. minimObj object, otherwise, gets set in the fileButton() method in this file.

filename2 - Default string for the titleSong object, the song played on the menu screen.
mainMenu, **inGame**, **gameOver** - Boolean flags used to determine what state the system is in and what actions should be taken. 3 main states, menu screen, in-game and game over screens.

1.2 Methods

setup() - Setup method, each processing sketch contains one. Used to set the size, graphics API to use and also creates the Background, MainMenu, newGUI and titleSong objects needed to start the game in the menu screen state for options to pick and start a new game

draw() - Draw method, also used once by a processing sketch, this is the main loop method. Sets up options for the graphics blending and determines the state of the system and what actions to take. If inGame, render the current state of the game object. If gameOver, render the MainMenu object in this state and if MainMenu, update the background visualisation, the gui controls and render relavant text to the screen (title).

stop() - Stop method, third main processing method, called when the system is exited. Uses relavant state to determine exactly which objects to close etc.

playButton() - Play Button method, called when the play button on the menu gui is called. Launches the game constructor after stopping the current song being played and updating the gui values also editable in-game. Lastly, loads in sfx objects if that option has been chosen and changes the state flags.

fileButton() - File Button method, called when the user clicks the fileButton and chooses a song to be played in-game. Sets the filename variable to the name of the file chosen. For safety, filters out all but mp3 and wav files as usable. Also uses ths swing component JFileChooser and the Runnable class to avoid conflicts between updating the file chooser and main draw method.

mousePressed() - Mouse Pressed Event method, used for left and right mouse click in-game. Left fires the player weapon and plays relavant sound if sfx on, also decreases the weapon energy value and sets a boolean value for firing to be true. Right sets a boolean value for whether the playing is pulling an object towards them (item drops) or not.

mouseReleased() - Mouse Released Event method used in conjunction with right clicking above, if released, attraction is set to false.

keyPressed() - Key Press Event - Used for moving back to the main menu, and for showing/hiding settings options and main HUD.

2) Asteroids

Author: Ashley Morrison
Class: Asteroid
Brief:

Class that represents the beating cubes that are the "asteroids" of the game. Contains a constructor, update, steer, HitPlayer, splitAsteroid, TextureCube, drawSide, render and drawRadarPoint methods.

This class uses the steer method taken from Daniel Shiffman's example boid class : http://www.shiffman.net/itp/classes/nature/week06_s09/seekarrive/Boid.pde

Asteroid objects are created and added to a list on a beat OnSet. The steering method is used to direct them towards the player. The objects are made up of a number of mini cube outer edges which move back and forth depending on the frequency spectrum value. More noise, cube sides move more etc.
The TextureCube, drawSide and the calling of drawSide in render() are all extended from another example : http://www.openprocessing.org/visuals/?visualID=2545

2.1 Variables

loc - Asteroid location vector.

vel - Asteroid velocity vector

dir - Asteroid direction vector, adjusted by the steer method and applied to vel.

collOffset - Asteroid offset value determined for the splitting of an asteroid into two, smaller versions. The offset is applied to the steer method direction to send the asteroids in opposite directions before coming back to the player.

r - Default value for the radius of the asteroid.

maxspeed - Float value used for the limiting speed of an asteroid calculated in the steer method.

maxforce - Float value used in a similar manner to maxspeed but applied at the end of the steer method.

sizeQuad - Used to determine how large the cubes that make up the asteroid should be spaced apart.

rotx - Rotates the asteroid around the x axis.

roty - Rotates the asteroid around the y axis.

prevTheta, theta, rotate, distance - All used for the drawing of the point on the radar corresponding to asteroid-player location discrepency. prevTheta and theta are the difference in dot products between the direction of the player and vector from player-asteroid from the last update/current update. rotateBy, the amount to rotate around the centre point.distance, the distance to offset the point from the centre corresponding to actual distance calculated.

radar1, radar2, radar3, radar4, goRight - Used to determine which of 4 states the radar is in, between increasing/decreasing past 180 and 0 respectively. The rotation needs to be different depending on which direction the player is heading in etc. goRight signifies whether the player is rotating roughly right or left and is also needed to update the radar correctly.

lives - The number of lives the asteroid has left.

start - A start value for use in the render method, calls to drawSide().

timer - Used for child asteroids created in hard mode to apply the collOffset value over a period of x.

timerDec - Used to decrement the timer value above.

spawnTimer - Used to determine how long the asteroid has been in existence and how long to leave it invulnerable to damage, only used for child asteroids. */

god - Boolean flag used with asteroids spawned due to a parent asteroid being destroyed, for a period, the child asteroid cannot be damaged.

parent - Boolean flag for whether this asteroid is a parent, if true & if difficulty=hard, spawnAsteroid() will be called.

removing - Boolean flag to check whether item has been hit and is being removed through creating an explosion/fading out.

isHit - Boolean flag to determine whether this asteroid has simply been hit and not yet reduced to 0 health.

colliding - Is this asteroid colliding with a missile object.

isDead - If colliding with player, set flag, it will be auto removed in the game loop.

alphaValue - Set to full for the life span of the asteroid except one removing=true, then this value is decremented to fade it out.

testExp - This asteroids explosion item to be called upon removing=true.

asteroidHealth - The int value of the health, decremented each time it is hit by a missile based on the number of points of the missile pattern.

2.2 Methods

Asteroid Ctr - Sets all major variables up, parent, sizeQuad, Start will all be based upon whether asteroid is a child or not etc.

update() - Used to add the offset & invincibility if relavant, to determine the direction to head in based on a target.

steer() - Creates a direction vector based on current position and target position.

HitPlayer() - Takes the position of the player p and determines whether the asteroid has collided

with the player based on the sum of the radius.

This method has been simplified from Daniel Shiffman's collision example:
http://www.shiffman.net/itp/classes/nature/collisions_s09/ballvsball_equalmass/Thing.pde

splitAsteroid() - Split Asteroid method called if the asteroid is a parent and if this difficulty is set to hard. This method removes this object after creating and adding two new asteroid objects to the list to be iterated through in MainGame. The two new asteroids are set with data based on the parent except with some values like lives and size decremented. The offset is also calculated to ensure the two asteroids move off in different directions to differentiate them visually for the player.

TextureCube() - This method is used to draw the asteroids to screen based on a size value passed in depending on parent/child.
It is based upon the MusicBox sketch: http://www.openprocessing.org/visuals/?visualID=2545

The difference being, only certain sides of the cubes are drawn depending on which side of the overall cube they are a part of. This is determined by the x value passed in. If 1, indicating it is the front face, draw following vertices.

Also, no textures are used with this version.

drawSide() - This method calls the above TextureCube in a translated position based on asteroid.loc

Based upon: http://www.openprocessing.org/visuals/?visualID=2545

render() - The render method starts the sequence of by calling drawSide in succession 6 times for the 6 faces of a cube. Before this the drawing is translated to the asteroid. loc vector, checks are made to adjust the drawing if being removed or if child (colour change). The rendering is rotated by incremented amounts in the x and y axis. Finally the start variable is used to how to access the sub bands of the frequency spectrum for each face of the cube so the sides move to the sound. This also depends on the size of the cube as a whole.

drawRadarPoint() - Method called each iteration in MainGame when in-game and not adjusting settings. Used to plot a point in relation to player to illustrate where the asteroid is in comparison. Due to the dot product between the player direction and player2asteroid vector returning between 0-180 (in front/behind), to adjust a point around 360 degrees in the correct manner, need to know the direction the player is heading in etc.

The radar variables represent this.

3) Background

Author: Ashley Morrison
Class: Background
Brief: This class represents the state of the main menu visualisation. It contains lists of SpiralGalaxy objects and points representing stars. The constructor adds a set amount to each list with a random x/y position on the screen.

3.1 Variables

spirals - Array list of SpiralGalaxy objects the background uses.

stars - Array list of points in space to draw stars at for background on menu screen.

3.2 Methods

Background() - fills the two array lists with SpiralGalaxy & PVector values respectively.

update() - Used to update the spiral galaxy objects based on whether the menu music beat is Onset. If it is, increment the colour, set the beat to true to be accessed in the SpiralGalaxy object render method. Adjust the twist value that determines the pattern of the spiral galaxy and finally if the number of points making up the galaxy is less than the max(20k), increment this amount by a random value.

render() - Draws each SpiralGalaxy and all star points to screen based on the current state of these values.

4) BeatListener

Class Taken exactly from minim audio library examples:
http://code.compartmental.net/minim/examples/BeatDetect/FrequencyEnergy/BeatListener.pde

Instantiated in the MinimAudio class for listening to different songs/types of beat.

5) Explosion

Author: Ashley Morrison
Class: Explosion
Brief: This class is used by each asteroid object upon being destroyed. It is based closely upon a point-spherical distribution sketch on OpenProcessing by 'Starkes':

http://www.openprocessing.org/visuals/?visualID=861

The explosion class constructor creates an array of Particle objects based on theta/u values which are passed in to the Particle object ctr to calculate a position around a sphere with them. The update method cycles through all particles and calls update and render methods on them.

5.1 Variables

numParticles - Number of particles to create around a sphere of radius x. */

radius, expansionInc - radius of sphere to distribute points and the increment for how quickly the set of points should expand in an explosion-like manner.

parent - Boolean value passed in from the asteroid that instantiated it. If true, explosion in 1 colour, else another.

Particles - set of Particle objects to update and render.

5.2 Methods

Explosion() - with parent boolean. Cycle through numParticles and create a particle with location around sphere.

update() - alpha value passed in from asteroid and further passed on to ExplosionParticle objects to use in render().

6) ExplosionParticle

Author: Ashley Morrison
Class: ExplosionParticle
Brief: This class is instantiated and contained within an arraylist of other ExplosionParticle objects to represent the explosion of an asteroid.

It is also based on the sketch at OpenProcessing:

http://www.openprocessing.org/visuals/?visualID=861
The explosionparticle class constructor creates sets up the relavant values passed in from Explosion. The update method updates alphaValue and calculates the latest x,y,z position of this particle. The expansion effect is done simply by incrementing the radius of the sphere about which the points are distributed. The expansion increment added to the radius value is itself also decremented meaning the radius expansion rate decreases over time.

6.1 Variables

theta, u - Values used to determine the x/y/z points in update()

x,y,z - X/Y/Z values for the position of the particle during the duration of the expansion.

tmpx,tmpy,tmpz - Previous x/y/z values for using to draw a line between old and current positions.

rad, expansionInc - radius and the expansion increment variable used to expand the set of particles around a growing radius size.

alphaValue - Alpha value passed in from Asteroid-Explosion to fade out after x period.

6.2 Methods

Explosionparticle() - theta, u, radius and increment values passed in initially.

update() - Changes the alpha value and recalculates the x,y and z values.

render() - Checks whether the particle is a part of a parent asteroid or not, changes colour values accordingly.

7) FFTSample

Author: Ashley Morrison
Class: FFTSample
Brief: Instantiated in the MinimAudio class. This class creates an FFT object from the minim audio library to extend the analysis that offers. It updates a current maximum and overall average on the FFT over the song to be used for the Rose/Missile classes for damage dealt/pattern generated.

7.1 Variables

song - AudioPlayer object from the minim library, used to play a song.

fft - FFT object also taken from the minim audio library for audio analysis.

currentMaximum - the current maximum peak value of the frequency spectrum.

average - the average frequency spec value calculated so far over the current duration of the song.

counter - the value used to calculate the average in the findMax method below.

7.2 Methods

FFTSample() - creates an fft object from minim based on the logAverages division of sub bands noted in the background of the thesis.

update() - forwards the fft sample on for analysis of the next frequency spec.

findMax() - Method used to calculate the current maximum peak and the overall average.

8) GUI

Author: Ashley Morrison
Class: GUI
Brief: Uses and contains all the gui related objects as well as update methods. Constructor creates just the menu items first then when the player hits the play button, the rest of the items are added due to being based on in-game values. Two different update methods for the in-game related data and the menu screen data.

8.1 Variables

file - File object for choosing a music file to play and retrieving the filename.

fc - file chooser object used with file to create a directory browser.

weaponEnergyBar,scoreBar,healthBar - Three main bar values for the player data. Used in-game to illustrate the game state from player's perspective.

gameData - The panel for the player data in-game.

gameSettings - The panel for the adjustable settings in-game.

mainmenu - The panel for the menu screen settings.

beatSensitivityBar, beatSensitivityBar2, noGalaxiesBar, noStarsBar, spaceRadiusBar - sliders to adjust the beat sensitivity, number of galaxies, stars and the radius of the sphere about which they're distributed.

wormholeNoStarsBar, wormholeRadius - slider for the number of stars in the wormhole and the radius of it.

galaxyNoParticlesBar, galaxyRadiusBar, asteroidSpeedBar, itemdropSpeedBar - sliders for the number of particles per Pulser object, the radius of them, the maxpseed value for asteroids and the same for item drops.

difficulty,difficulty2,mode,hud,sfx,sfx2 - Radio buttons for the difficulty, game mode, hud being on/off and sound effects being on/off.

vol1, vol2 - Sliders to adjust the volume of the music for the menu and in-game songs.

filename - a string variable to be used for holding the filename that is chosen by the user to play in-game, passed back to Main, then MainGame and finally game.minimObj.

8.2 Methods

GUI() - instances all the menu screen gui related items and adds them to mainmenu ControlP5 object.

createGameGUI() - method called after the play button above has been pressed. This instantiates the in-game and player stats objects and adds them to the respecive controlP5 object.

update() - Updates the in-game player stats as well as the in-game settings.

update2() - Updates the main menu settings.

9) ItemDrop

Author: Ashley Morrison
Class: ItemDrop
Brief: Class that represents an item drop/player stat booster in the game. Item drops are spawned from the centre of Pulsers and can be one of three types corresponding to the three types of pulsers. These three types are displayed through the colours red, blue and yellow which if retrieved by the player boost the score, weapon energy level and shield respectively.

Once spawned they use the steer method taken from Daniel Shiffman's example boid class: http://www.shiffman.net/itp/classes/nature/week06_s09/seekarrive/Boid.pde

Using this they travel towards the centre of the wormhole which is at an arbitrary point around the player. The player has the option of attracting the item towards him/herself by way of the right mouse button and diverting it in their direction. If the itemdrop effectively collides with the wormhole it is removed, if it collides with the player then the relavant boost will be awarded after which it's removed.

The spawning of the item drops is designated by a particular type of Onset being detected (either a kick, hat or snare) along with the type of pulser matching this onset detection being compared. Along with the ctr and steer methods are update, render and drawRadarPoint methods.

9.1 Variables

pos - A vector representing the position of this item.

dir - A vector representing the direction of this item.

target - A vector representing the target, whether the wormhole or player.

attraction - A boolean for whether the player is currently diverting the item to them-self.

isDead - A boolean for whether the item has collided either with wormhole or player and can be removed.

maxspeed, maxforce, origMaxspeed - Floating point values for the maximum speed and force this item can accelerate to. The maxspeed can be altered when the player is attracting it towards them, hence the use of an original value for when this is no longer the case.

prevTheta,theta,temp,distance - All used for the drawing of the point on the radar corresponding to asteroid-player location discrepency. prevTheta and theta are the difference in dot products between the direction of the player and vector from player-asteroid from the last update/current update. rotateBy, the amount to rotate around the centre point.
distance, the distance to offset the point from the centre corresponding to actual distance calculated.

radar1,radar2,radar3,radar4, goRight - Used to determine which of 4 states the radar is in, between increasing/decreasing past 180 and 0 respectively. The rotation needs to be different depending on which direction the player is heading in etc. goRight signifies whether the player is rotating roughly right or left and is also needed to update the radar correctly.

type - An integer to represent one of three types the item can be which dictates when/where it will

be spawned and the colour/shape deformation as well. Either 0, 1, or 2, i.e. gold, red or blue, i.e. shield, score, weapon modifier.

9.2 Methods

ItemDrop() - The item drop constructor, all values passed in, including type which is determined by the type of onset detected and that matching the pulser type being compared. Direction updated through steer method in update()

steer() - Creates a direction vector based on current position and target position.

Method directly taken from Daniel Shiffman's example boid class :
http://www.shiffman.net/itp/classes/nature/week06_s09/seekarrive/Boid.pde

With the additions of a boolean flag when the distance between asteroid-player is not greater than 0 and what to do if this is the case, i.e. awarding of points, shield or weapon bonuses if the collision is with the player.

update() - The update method for calculating the distance between the item and target. If attraction is set the target is the player, if not, the wormhole.If the item is being attracted by the player, the maxspeed is adjusted as twice the original value. If not, the maxspeed is proportional to the average value calculated in FFTSample meaning the difficuly in retrieving the items fluctuated with the development of the song. The direction vector is passed to the steer method and then added to the position.

dec is the average passed in from MainGame that is taken from FFTSample.

render() - Draws the item as a low detail sphere with a trail of smaller spheres behind it by offsetting in the opposite direction of the item drop.

drawRadarPoint() - Method called each iteration in MainGame when in-game and not adjusting settings. Used to plot a point in relation to player to illustrate where the asteroid is in comparison. Due to the dot product between the player direction and player2asteroid vector returning between 0-180 (in front/behind), to adjust a point around 360 degrees in the correct manner, need to know the direction the player is heading in etc. The radar variables represent this.

10)MainGame

Author: Ashley Morrison
Class: MainGame
Brief: This is the game object that contains the game loop with lists of all asteroids, item drops, missiles and galaxies in-game. Stars, galaxies and asteroids are spawned at varying offsets from a sphere, the radius of which was set in Main. This class also contains an instance of MinimAudio for the playing/analysing of the in-game song used for spawning asteroids and item drops.

It also contains camera rotation code for rotating around a point in 3d, HUD code for displaying the current key positions of asteroids/items etc, player scores and a rose/weapon pattern as well as in-game settings that are changeable by hitting TAB. The HUD can be hidden with SHIFT.

Lastly it uses the Traer Physics library to keep hold of a set of particle position fired in some direction that is used to map a pattern of dots to representing the player attack. The bulk of this code is found in the Rose/Missiles classes respectively.

The loop checks for whether it is in setup mode or not, if setup, simple display a blank screen with the in-game settings, else cycle through all the objects in the game and update/render.

10.1 Variables

minimObj - The MinimAudio instance used to create, play and analyse the song of choice for in-game. If in-game and not setup, play.

onSet - Used as a flag for when there is a onset detected and if this matches with the firing of the player weapon, 2 extra missiles are fired giving a boost to the play for synchronised efforts.

pulsers - The array list of pulser objects created in the constructor with positions around the player. Updated in the Pulsers & updatePulsers methods respectively.

asteroidList - Array list of asteroid objects to be cycled through each update and calling each asteroids update method.

missiles - Array list of missile objects in the game at any given moment, missiles objects have a finite life span and are periodically removed in the update methods found in Missiles()

starsAlpha - the alpha values for drawing the points of the stars around the sphere to give a greater feeling of depth in the scene.

stars - The corresponding stars array which uses the alpha values above and is an array of positions around a sphere used to draw points to.

galaxies - 2d Array of floating point values for the distribution of Pulsers around the player, at an offset from the normal spherical distribution.

cam - The Matrix used for camera rotation found in CamRotation()

gravitate, isFire - Boolean flags for whether the player is attracting an item to his/herself and whether the player is firing the weapon.

newRose - Rose object which is used to generate patterns that are connected with the current max

and averages calculated in FFTSample that are used to display the missile objects. A preview of the current iteration is shown on the HUD.

n - int that is updated to the overall average calculated in the FFTSample object and passed to Rose.setPoints.

mx, my - Mouse x and Mouse Y values updated in the main loop.

d - The direction the player is facing.

tmpDir - non normalized player direction vector.

pos - The position of the player at 0,0,0.

physics - The particle system including from the Traer Physics library: http://www.cs.princeton.edu/~traer/physics/

q - A particle to be re-assigned and attached to missile objects each time one is spawned.

closeAsteroids - An array list of the asteroids that are within a certain range of the player to be used for drawing the closest asteroids on the radar.

nTarget - Is the player currently closely lined up with either an asteroid or an item drop, if yes, true.

onAttract - Is the player currently onTarget, holding down right mouse button and has enough weapon energy to attract an item.

tmpxRot - Amount of rotation in the x axis, used in CamRotation()

tmpyRot - Amount of rotation in the y axis, used in CamRotation()

items - Array list of all the item drops currently running in the game.

itemOK - Boolean variable periodically set to allow the onset of a beat to result in items being spawned, to restrict how often items are spawned.

itemCounter - Integer used with itemOK.

addItems - Integer counter used to total up how many items are being added in any one onset detection, again for limiting purposes.

newWormhole - Wormhole object that is the point of attraction for all item drops.

centreWormhole - A vector showing the centre position of the wormhole, used for the exact target item drops move towards.

backCol - Used for the background colour, fades to white on completion of a song.

playerScore, playerHealth, playerWeaponEnergy, playerWeaponEnergyBase - Integer score and shield values, floating point weapon energy values, base for when weapon energy adjusted.

maxspeedAsteroids - Default value for the maxspeed asteroids can travel. Can be changed in the

settings in-game.

maxspeedItems - Default value for the maxspeed item drops can travel at. Also editable in the in-game settings.

setupGame - If the player presses TAB, settings gui appears and main game if statement is re-directed to a blank screen and gui, otherwise, update,render as normal.

gameDifficulty - An integer for accessing/updating the game difficulty, which can be changed in the settings.

wormholeCentrePulse - The pulser object drawn at the centre of the wormhole object.

10.2 Methods

MainGame() - The constructor for the game object. Constructs wormhole, item list, Rose object, particle system, stars alpha value, stars, galaxy positions, asteroidList, pulser positions, missile list, camera matrix, minimObj & the Pulser object at the centre of the wormhole in this order.

updatePulsers() - Method used to update pulsers particle positions and check for beat onset detection and spawn item drops if necessary. Called from Pulsers() method below which cycles through galaxy array and calls this method with the current index value.

CamRotation() - gets the amount of rotation in the x/y and rotates the current matrix by these values before finding the direction vector and calling the camera() method with d.

Asteroids() - Method called in main game loop render() to cycle through, update, add and draw asteroids to the game.

Pulsers() - Cycle through galaxy array positions, translate and call updatePulsers to detect beat onsets, spawn items and draw accordingly.

Wormhole() - change the wormhole rotation speed dependent on the average of the FFTSample, increment the colour value. Translate and draw the pulser object to the centre of the wormhole.

Items() - Cycle through array list of items. Same calculation for items and onTarget as for asteroids. Also check if right mouse button down and weapon energy available before diverting item with boolean flag switch.

Stars() - Cycle through stars array, translate to position and draw a point using the alpha value of the same index to adjust brightness.

Missiles() - Cycle through all current missile objects, check whether player has energy before spawning more and check if firing coincides with a beat onset and if so, fire two extra missiles for the price of one as bonus.

HUD() - draws crosshair, rose pattern and radar points for asteroids, item drops and the wormhole.

render() - main game loop. Determines whether in setup mode or not and what to draw otherwise. Updates gui values, determines when to change to game over state and in what fashion. Also replenishes weapon energy, increments item counter to limit number spawned, updates the audio object and mouse positions along with calling all the object specific methods above.

11) MainMenu

Author: Ashley Morrison
Class: MainMenu
Brief: This is a class that draws text to the screen depending on what state the system is in, either title or game over.

12)MinimAudio

Author: Ashley Morrison
Class: MinimAudio
Brief: Encapsulates all the audio related objects and method calls that the system uses. Contains minim library objects: Minim, AudioPlayer, FFT, BeatDetect and instantiations of the BeatListener class. Uses three different beat listeners for the menu screen song, the in-game soung energy detection(asteroids) and frequency detection(pulsers). Sets two main values, the overall average frequency value and a latest maximum peak value.

12.1 Variables

minim - Main Minim audio library class object.

song - Minim library object for loading in/playing songs.

thisSample - instantion of the class which encapsulates and uses the minim library FFT class.

bl,beat, bl2,beat2, bl3,beat3 - Beat detection and listener objects for the three varities of beat detection used.

currentNoDots - set in the update method to be proportional to the current maximum peak value.

sLength - song length in milliseconds.

tempFactor - used to return the fraction of the song completed which in turn is used to blend the pattern of the wormhole from a starting value to an end design.

12.2 Methods

MinimAudio() - Takes in the PApplet to use for constructing the Minim object. Also a boolean value to determine whether the state is currenty in the title screen or not.

update() - updates the FFT which forwards the analysis of the spectrum values. Recalculates the factor by which to adjust the wormhole pattern. Also calls findMax to calculate the maximum peak and overall average values.

13)Missile

Author: Ashley Morrison
Class: Missile
Brief: Missile object which takes the pattern generated by the Rose class for display and the position returned by the particle in the Traer particle system as it's position upon being generated and fired in the direction the player is facing. Contains an update method which checks for collisions and renders the pattern to a new point in space based on a particle position. Each missiles has it's own particle object which is updated by the "physics.tick()" call in MainGame. Also has a timer value that is decremented resulting in missiles being removed after x time if they haven't hit anything.

13.1 Variables

r - radius of missile object.

timer - Timer value decremented over time resulting in time span before removal.

p - This missiles particle object, get updated in MainGame through physics.tick call.

rotx, roty, rotz - rotational values for displaying the pattern in front of the player.

maximumSpecValue - the value taken from the FFTSample for currentMaximum when this missiles was created.

averageSpecValue - the value taken from the FFTSample for the overall average when this missile was created.

13.2 Methods

Missile() - constuctor - passes in the particle object from MainGame as well as rotational values and the currentMaximum/average FFT values.

update() - Method to update location, increments the z rotation, checks for collisions, reduces timer and renders to screen if still alive.

collision() - Gets the particle position, compares with asteroid positions in the same way the asteroid collision detection does using Daniel Shiffman's example collision code: http://www.shiffman.net/itp/classes/nature/week06_s09/seekarrive/Boid.pde

render() - Method to display missile.

14)Pulser

Author: Ashley Morrison
Class: Pulser
Brief: This class is heavily based upon the OpenProcessing example 'Gravity Swarm' by Claudio Gonzales: http://www.openprocessing.org/visuals/?visualID=2363

The Swarm.pde file is now the Pulser class and the particle.pde is now the PulserParticle class. Besides an initial call to gravitate the particles around a central point in the ctr, each pulser has it's array of PulserParticles updated/displayed through calls made in the MainGame.updatePulsers() method depending on beat onsets. It has also been adjusted for OpenGL, and each Pulser object is distributed in 3d space around a position offset from a sphere.

Also, the initial distribution of the PulserParticles is extended to be around a sphere as this influences the type of pattern to be made subsequently. This distribution around a circle uses a 2d version of: http://www.openprocessing.org/visuals/?visualID=861 by 'Starkes'. The point of attraction is also designated as the centre of the circle.

14.1 Variables

Z - Array of PulserParticles, size set in the title screen.

pos - Positional vector of the Pulser.

avgX, avgY - x,y value to gravitate around.

type - The type of the PulserParticle, randomly chosen at the MainGame setup to be one of three types used in conjunction with the item drops and beat detection.

14.2 Methods

Pulser() - Ctr method, passes in an integer for the type and a boolean designating whether this Pulser is a generic one or the one used with the centre of the wormhole.

15)PulserParticle

Author: Ashley Morrison
Class: PulserParticle
Brief: The particle class to be used with the Pulser objects. Contain x,y,z values and previous x,y,z values as well as magnitude, angle and mass floating point values for use in pulling them towards a position in a way analogous to the pull of gravity. If this is the first time instantiation, calculate the x/y positions around a point on a circle. Else x/y = theta/u.

15.1 Variables

x,y,z,px,py,pz - Current and previous x,y,z values.

magnitude - floating point value for the magnitude of the pull.

angle - floating point value for the angle of attraction.

mass - floating point value for the mass of the particle which will determine the magnitude of the force.

theta - theta value to be passed in from PulserParticle.

u - u value to be passed in from PulserParticle.

radius - radius of the distribution of the particles, set in menu screen.

15.2 Methods

PulserParticle() - passed in x/y values, magnitude, angle and mass.

gravitate() - Takes in a new PulserParticle object to be attracted to which has the position of being in the centre of the circle. Uses equations for calculating the magnitude and angle values taken from: http://www.openprocessing.org/visuals/?visualID=2363

display() - decreases the magnitude value to slowly bring the particles to a stop over time. updates the x,y values based on the magnitude/angle values calculated in gravitate() and draws a line between the current and old x/y positions before updating the old ones.

findAngle() - Takes in the x/y values for the Particle to be attracted to. This method is unaltered from the original example: http://www.openprocessing.org/visuals/?visualID=2363

16)Rose

Author: Ashley Morrison
Class: Rose
Brief: This class is a modification and re-use of Jim Bumgardner's 'Rose Display' OpenProcessing sketch: http://www.openprocessing.org/visuals/?visualID=1555

It's also the sketch that most inspired the possibilities of using audio to fuel visualisations for this project. The key value for the equation that is documented in the background section of the thesis is a value n which determines the number of petal like shapes or curves the pattern generates. This rose generation now takes in a value n that is connected with the FFTSample class and the average of those frequency values. It also passes in a number of points to be used for drawing the pattern which is the current maximum peak value, hence in a moment of explosion in the audio, the rose generation will jump in complexity.

16.1 Variables

n - set in MainGame to be equal to the average+5. */

theta - used to calculate the radius, colour and curve points.

sizePoint - how big should the points be displayed as.

rr,gg,bb - the red, green and blue values selected in the setDayglowColor() method, left from the original sketch.

16.2 Methods

Rose() - Constructor.

setPoints() - Takes in a radius value to determine the size of the pattern drawn, a new n value, the number of dots to draw and a rotational value.

setDayglowColor() - takes in a hue value which is also proportional to the n value having been passed in from setPoints()

17)SpiralGalaxy

Author: Ashley Morrison
Class: SpiralGalaxy
Brief: SpiralGalaxy class modified slightly from Philippe Guglielmetti's 'Spiral Galaxy' OpenProcessing sketch: http://www.openprocessing.org/visuals/?visualID=699

This class is used for the menu visualisation and contains modifications the the display of the points as well as allowing the number of points to be added to. The result on the menu screen is a set of spiral patterns that develop with some synchronisation to the music, with more points being added over time as well as changes in colour etc.

17.1 Variables

colour - the colour of the spiral galaxy to be used.

stars - the current number of stars, incremented over time with the audio.

Rmax - galaxy radius.

speed - a random rotational speed.

eratio - ellipse ratio.

etwist - twisting factor (orbit axes depend on radius)

angle - Array list of angle values.

radius - Array list of radius values.

angleArray - array of angle values.

radiusArray - array of radius values.

cx, cy - centre x and centre y values.

beat - Boolean flag for whether the corresponding beat has been detected.

colourChange - Boolean flag for whether it is time increment the colour. Changed in Background class.

pos - Position of the spiral galaxy in 2d on the screen.

17.2 Methods

SpiralGalaxy() - Constructor, takes in a radius size. Initialises the star angle/radius array values to be used in determining point positions in render().

updateStarCount() - Convert between the array and array lists to adjust the number of stars to draw per galaxy.

drawGalaxy() - cycles through angle and radius values to calculate latest x,y values around centre

point. Slightly modified to adjust for star count changes and colour changes.
http://www.openprocessing.org/visuals/?visualID=699

18)Wormhole

Author: Ashley Morrison
Class: Wormhole
Brief: Wormhole class modified from Philippe Guglielmetti's 'Spiral Galaxy' OpenProcessing sketch: http://www.openprocessing.org/visuals/?visualID=699

This class is used for the wormhole feature in-game that item drops are drawn towards. The changes made are setting the centre point, pulsing the colour of object drawn, altering the appearance by not drawing X points that are close to the centre to give more of a black hole effect, changing the ratio/twist values determining the pattern drawn when there is a beat and by a factor of how far along the song is. The wormhole position is also drawn to the HUD to relay it's position in relation to the player.

18.1 Variables

speed - a random rotational speed.

eratio - ellipse ratio

etwist - twisting factor (orbit axes depend on radius)

angleArray - array of angle values.

radiusArray - array of radius values.

cx, cy - centre x and centre y values.

x,y - values to translate the wormhole by.

centre - A Vector representing the centre position of the wormhole points.

prevTheta,theta,temp,distance - All used for the drawing of the point on the radar corresponding to asteroid-player location discrepency. prevTheta and theta are the difference in dot products between the direction of the player and vector from player-asteroid from the last update/current update. rotateBy, the amount to rotate around the centre point.distance, the distance to offset the point from the centre corresponding to actual distance calculated.

radar1,radar2,radar3,radar4, goRight - Used to determine which of 4 states the radar is in, between increasing/decreasing past 180 and 0 respectively. The rotation needs to be different depending on which direction the player is heading in etc. goRight signifies whether the player is rotating roughly right or left and is also needed to update the radar correctly.

beat - Boolean flag for whether the corresponding beat has been detected.

hole - floating point value to determine the size of the black hole that is incremented with a beat detected.

colour, increasing, decreasing - A colour value in the hsb range and two boolean values determining whether the colour is being incremented or decremented as it phases between almost invisible and full brightness when a beat is detected.

18.2 Methods

Wormhole() - Constructor for the Wormhole class.

calculateCentre() - uses the index found above for the value in the radius array that was smallest to access this in working out the x/y values and setting centre accordingly.

colourInc() - Used to set a colour value for drawing the points, flips between increasing and decreasing the value to create a slow pulsing effect and changes to full brightness on a beat detection. Uses hsb scale, maximum value of 0.4 before decreasing to near 0.

render() - cycles through angle and radius values to calculate latest x,y values around centre point. Slightly modified to adjust for changes in the pattern and black hole size.
http://www.openprocessing.org/visuals/?visualID=699

drawRadarPoint() - Method called each iteration in MainGame when in-game and not adjusting settings. Used to plot a point in relation to player to illustrate where the asteroid is in comparison. Due to the dot product between the player direction and player2asteroid vector returning between 0-180 (in front/behind), to adjust a point around 360 degrees in the correct manner, need to know the direction the player is heading in etc. The radar variables represent this.