AG0801A "Console Game Development" Coursework 2 Report

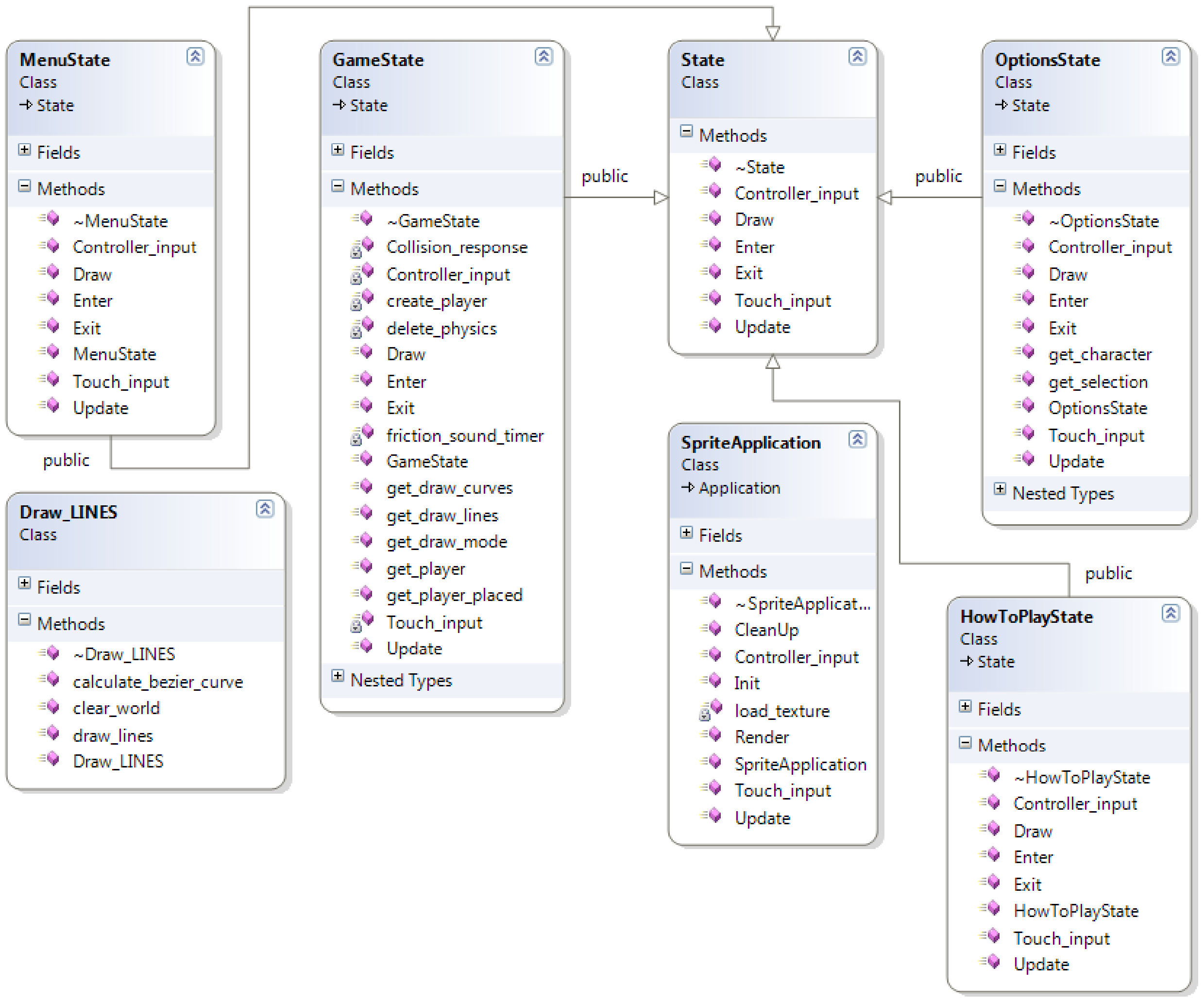
Introduction

Semester 2 saw Console Game Development students take on designing and building an application for the PlayStation Vita. The application was to utilise functions and features taught in class and implement functionality of the Box2D physics engine. Through this, the application would be a single-level playable demonstration of the game created and showcase the mechanics involved.

The application created grants the user control over the level design. After choosing the character and vehicle, the player is required to design a physics playground by drawing lines and curves for their chosen character to slide along. Using straight lines and curves the player has the ability to draw to their hearts desire to watch as gravity takes hold and their character tumbles down the slopes. In order to give the player the most fluid control when drawing, the touch-screen functionality of the Vita has been put into full effect. The player will use this not only to draw but also navigate menus and be given visual feedback for what drawing mode they are in and any given on-screen button presses. The application design and the techniques used will be discussed throughout this report. Alongside this, a user guide explaining how to run and use the program will be provided as well as critical analysis of the practices implemented.

Application Design

The design stage of an application is by far one of the most important. It can single-handedly rout out any discrepancies in thought-processes and class structure and layout. The coursework, as with any worthwhile project demanded good structure. Thus, classes were the first thing to tackle. Several classes have been used in the application; 'GameState', 'OptionsState', 'MenuState', 'HowToPlayState', 'GameObject' among others past the given 'SpriteApplication' class. This allowed for an easy collaboration of files to link everything together and get the application running smoothly.



As it can be seen from the diagram, the four main classes; ‘MenuState’, ‘GameState’ and ‘OptionsState’ and ‘HowToPlayState’ all inherit from the base class ‘State’. This base class contains virtual functions with no definitions allowing classes inheriting from it to create their own individual versions. ‘GameState’ takes the base class and expands on it to give the game more functionality. This class also makes use of the ‘Draw_LINES’ class to create the players level when in game. The functions outlined in the diagram have values passed to them in the ‘GameState’ class and are then able to add physics and sprites to the gameworld. Furthermore, the ‘SpriteApplication’ class takes all previously mentioned classes and allows them to run. Through the use of pointers, the class is able to switch between game states and this is where the different classes come in. Each class inheriting from ‘State’ contains the relevant information for each state in-game and ‘SpriteApplication’ is able to access what is needed to be able to update the game, render each frame and switch states when necessary.

Lastly, Box2D presented an opportunity to use an engine for the first time. It was then imperative to utilise it properly. In order to accomplish this a new header and .cpp file were created to hold any necessary body-creating functions to be used within the game and possibly used at a later date. Box2D allows for the creation of physics and simulates the reaction between the created bodies and external forces in the world. This opens up a considerable number of options when considering the applications final purpose. Culture hits such as 'Angry Birds' and 'Paper Toss' use physics to great effect with the former using Box2D itself. Line Rider, the applications' main inspiration uses flash as its base and uses a similar physics simulation. Thus it was relatively simple to replicate some of the desired effects necessary for the application and create something with satisfying polish and finish the application needed.

Techniques Used

The application was required to accomplish certain things with its creation. These included; a menu system allowing the player to alter the game in some way, appropriate use of the PlayStation Vita controls, the use of Box2D to provide a physics simulation and utilise sound within the game.

The in-game menu takes a simple approach to altering the game by changing enum states to dictate what happens within the application. Where many games of the modern era will allow players to change difficulty, the number of players and sound volume among others the application created takes a different approach. Two options are available to the player; changing the texture of the sprite to alter the appearance of the player character and also the vehicle used within game. The latter especially can have a drastic impact on the simulation since it changes the way the player thinks about the level they create. For example, if the player was to choose a car over a hamster-ball, then they would be able to draw ramps and uphill climbs for the physics simulation to tackle whereas the hamster ball requires gravity and a downward slope to get it moving. Within the game itself, the player has choices on how to draw the level they create also. Much like MS Paint they will find a line tool for drawing straight lines and a curve tool to draw curved lines. On top of that they will find an eraser for rubbing out mistakes and a bin tool which will clear the screen of sprites and physics.

The aforementioned curved line tool uses a mathematical system for drawing very smooth graphical curves. Bezier curves take a pre-determined number of points (in this case 5), the first and last are anchor points where the curve will begin and end respectively. The remaining points are control points which bend the curve towards them. However, unlike B-splines, the curve does not pass through the control points but still delivers a very satisfying curve. When the curved line tool is selected, the player is required to touch the screen in the desired five points and the program will draw a Bezier curve to match.

In order to achieve a natural feeling and fluent control scheme for not only the curve line tool but almost all controls within the game, the PlayStation Vitas touchscreen has been used to full effect. The player is faced with this control system from the moment the game loads - using their finger to navigate menus and start the game itself and once in the game to draw and develop their level. On top of this, when the car vehicle has been chosen from the options menu the controller buttons are used to manipulate it's acceleration.

One of the most fundamental parts of the coursework requirement was the application of the Box2D physics engine. The simulation was to be used to create the motion of the sprites on-screen and any collision detection needed therein. The game uses this to create physics behind the players created level by implementing Edgeshapes to prevent the player falling through their creation. This combined with gravity provides a really nice effect for the player to enjoy. By creating a downhill slope and a loop-de-loop the physics simulation can carry the chosen vehicle down and round the loop with no input needed from the player at all. This is in essence the idea behind the game and with Box2D the physics are simulated perfectly.

Lastly, sound effects and music were needed to add the final touch to the game. There are several assets used in the application and are employed through different methods. The initial splash screen and following menus carry the games theme song on loop until the player decides to start the game. Following this, a new, calmer song is played to encourage creativeness and keep the player happy. When the player starts the simulation and their chosen vehicle hits the level a looping sound effect is played; an engine noise for the car and a soft friction noise for the other.

User Guide

In order to run the application, it is necessary for certain files to be present. In the designated 'bin' folder; 'buttons.png', 'game_background.png', 'game_buttons.png', 'title2.png', 'car_body.png', 'credits_background.png', 'howtoplay_background.png', 'Mouse_in_a_ball.png', 'options_background.png', 'options_buttons.png', 'penguin_ball.png', for textures and for music 'bensound-popdance.wmv', 'bensound-psychedelic.wmv', 'Race Car Idle-SoundBible.com-804330831.wmv', ‘Rustle-SoundBible.com-1736422480.wmv’, ‘91926__corsica-s__ding.wmv’ and the given ‘comic_sans.fnt’ and ‘comic_sans.png’.

Past this, in the main project folder all required .cpp and corresponding .h files are needed; ‘Box2D_Bodies’, ‘Draw_LINES’, ‘game_object’, ‘GameState’, ‘main.cpp’, ‘MenuState’, ‘OptionsState’, ‘HowToPlayState’, ‘sprite_application’ and finally ‘state.h’. So long as all these files and the obvious PlayStation Vita development kit are present, the program should run as desired.

As mentioned previously, the touchscreen on the Vita has been used to manipulate the game. Navigate menus, change drawing modes and draw the desired level by touching the screen. To draw a straight line, select the Line tool and swipe the screen to draw a line from the first-touched position and the last. In order to draw a curve, select the Curved line tool and touch the screen in five successive points. This will draw a curve from the first point to the last and the remaining three shall manipulate the curve towards them. Lastly, to control the car it is a simple case of using the ‘X’ button to accelerate forward and the ‘Square’ button to reverse.

Critical Analysis

The application created fulfills all requirements of the coursework specification. It uses 2D sprites to display the main menu and buttons, the player character and the lines the player draws. Alongside this, sprite animation is used to not only add a finishing touch to the buttons by allowing them to depress into the background but also to give life to the player character. As previously mentioned, a menu system has been employed to allow the player to change certain aspects of the game - in this case the player is able to choose between vehicles and character textures.

Box2D has been implemented to give gameplay functionality and a physics simulation to manipulate the character motion. Edgeshapes from the Box2D library have been used to form the lines drawn by the player and give the players vehicle something to collide with. Gravity is used to full effect to provide motion to the character as it slides down and along the level created.

The PlayStation Vita controls have been used in a such a way to provide the player with as much control as possible. The touchscreen allows the player to navigate menus and draw their level with little effort. On top of that when the car is selected from the options menu, the controller pad is used to give extra control and level variation by applying forward and reverse acceleration. Finally, sound effects and music have been added to round off the game and give it some polish.

When it comes down to improvements for the application, there are a few areas which could be developed. Firstly, a canvas could be introduced along with proper camera movement to give the player a greater area to create their level whereas currently they are confined to the standard screen size of 960x544 pixels. Furthermore, more vehicles could be added to allow the player to see what could happen with the different combinations. Games such as Happy Wheels do this to great success and many players get enjoyment out of seeing the vehicles react differently to the levels put in front of them.

For the player to have a slightly more natural control scheme, b-splines would be incorporated over Bezier curves. The difference between the two are the control points. Bezier curves as mentioned previously simply manipulate the curve toward the control points but the curve does not pass through them which can be confusing to the player. B-splines change this and forces the curve through the control points, making for a more fluid form of curve drawing for the player.

However, the Bezier curves are certainly a highlight to the program as well. The mathematics behind the Bezier method is complicated and has been implemented in the simplest form possible. By splitting the x and y components of each anchor and control point of the curve, it was possible to apply physics and sprites to each segment of the curve. Through this, each curve adds a significant number of objects for the application to deal with. A standard Bezier curve requires an iteration from 0 to 1 by increasing an included variable by 0.01 and thus creating 100 objects. In order to reduce this and therefore reducing the amount of lag, the variable is instead being increased by 0.05. This strikes an even balance between performance and curve smoothness.

A further limitation of the program and certainly a future improvement is the deletion of straight lines. Currently, there are two ways to delete the lines on screen. Use the Bin tool to clear the screen entirely of sprites and physics or by using the eraser tool. Where the eraser tool works perfectly when erasing curves, due to the nature of how they are formed, the same cannot be said for straight lines. When the eraser passes over a straight line it will completely remove it from the game. In future, it would drastically improve gameplay if when the eraser crossed the line it instead removed only to the width of the eraser, in essence creating two new lines in its wake on either side of where the eraser hit the line. This would give the player far more control of their scene and allow for more intricate design.

Finally, a level save feature could be implemented to allow the player to save their creations. Whether to share with friends or simply to edit them at a later date. Currently, when the player is in game they are unable to change the character or vehicle chosen in the options menu. By being able to save what they are working on it would be very easy to change vehicles, load up their saved level and watch a new collection of physics attempt their creation.

Conclusions

In conclusion, this coursework has been very informative and enjoyable. The addition of the Box2D physics engine proposed a new and welcome challenge and opened up a new raft of game mechanics which could be easily implemented. Alongside this, utilising new features of the Vita such as the touchscreen and music capabilities again added a new dimension of playability and polish to the final application.

Furthermore, despite having learned enough in class to cover the coursework material each coursework is what you make it. This coursework saw to push the boundaries and the application turned out better for it. Box2D and the PlayStation Vita offered an interesting platform to create something new. The mathematics involved in the application too meant a learning curve and created a nice effect in the end.

In future, more planning should be involved in the beginning stages of the application and thus would make any problems faced be easier to tackle. If this coursework was to be done again, it would be a benefit to research all aspects of Box2D to maximise the knowledge necessary before tackling such an ambitious project. Despite this, the game in question is to a high standard and meets the brief without question.

References

Malin Christersson. 2014. *De Casteljau's Algorithm and Bézier Curves.* [online]. Available from: http://www.malinc.se/m/DeCasteljauAndBezier.php. [Accessed on 15th June 2014].

N-Elton-Crew. 18th July 2011. *transformice_mouse_sprites_by_waails_the_angry_fox-d40f4oq.png.* [Online image]. Available from: http://n-elton-crew.deviantart.com/art/Transformice-mouse-sprites-242570618. Edited to make 'Mouse_in_a_ball.png'. [Accessed on 27th June 2014].

Ford, C. 24th August 2009. *Rustle-SoundBible.com-1736422480.wmv.* [Sound recording]. Available from: http://soundbible.com/895-Rustle.html, with License Attribution 3.0 (https://creativecommons.org/licenses/by/3.0/). No edit. [Accessed on 19th April 2014].

Bensound. 2012. *bensound-popdance.wmv.* [Sound recording]. Available from: http://www.bensound.com/royalty-free-music/track/pop-dance, with Royalty Free license. [Accessed on 19th April 2014]

Bensound. 2012. *bensound-psychedelic.wmv.* [Sound recording]. Available from: http://www.bensound.com/royalty-free-music/track/psychedelic, with Royalty Free license. [Accessed on 19th April 2014].

Corsica_S. 10th March 2010. *91926__corsica-s__ding.wmv.* [Sound recording]. Available from: https://www.freesound.org/people/Corsica_S/sounds/91926/, with License Attribution 3.0 (https://creativecommons.org/licenses/by/3.0/). No edit. [Accessed on 19th April 2014].

SoundBible.com. 15th July 2009. *Race Car Idle-SoundBible.com-804330831.wmv.* [Sound recording]. Available from: http://soundbible.com/738-Race-Car-Idle.html, with Personal Use Only license. [Accessed on 4th May 2014].