SPLOX RACERS

By Kate Sarah Boehner

PROGRESS REPORT

FOR UNITS:

Game Development Project – CGP601

Version 1

Contents

1. Introduction

1.1 Background

Splox Racers is a 3D sandbox racer created as the major project of the Southampton Solent University. The tracks will be generated on splines. Those splines can be created by the developer or a random generator and can be saved to the disk to load at a different time. The AI uses the splines as a guideline to figure out, how to drive along the track. Different points on the spline have different attributes to make different areas, like boosters or obstacles, on the track available. Each point also has attributes to control the tracks banking, thickness, width and whether or not it has guard rails available.

The players can build their own cars via a sandbox editor. There will be different block shapes available to create the chassis in any form. Blocks can be coloured to further improve the look of the car. Special blocks will be needed to change the cars behaviour, like speed, handling and so forth.
The editor makes sure, that the car has all necessary parts, exactly one seat, wheels and an engine.
Players can test their cars in a special track or a chosen/generated track to help fine-tune the cars behaviour.
If they are happy with the car, they can name it, save it and choose to drive it in races.

As of my research, no one ever combined a random spline track generator and a vehicle builder into a racing game.

1.2 Challenges

There will be quite a lot of challenges to overcome.

The mesh of the track has to be generated, so there are no holes or other artefacts in it and all the textures line up without any seems.

The random generator has to generate valid tracks by making sure the spline does not overlap itself and keeps enough distance, so that cars will not be blocked by the track. Loops have to be placed in a way that the players can build up enough speed to be able to make it through the loop. Obstacles have to be placed in a way that makes them fun i.e. make sure the distance between obstacles is not too low, as that will make the track boring, and not too high, as that will make the track too frustrating.

The car editor has to be easy to use and make sure the build cars are always valid. Therefore, it has to check for disconnected blocks and indicate them to the player. It has to make sure the player places all the mandatory parts and constraint the placement of blocks to keep the cars size to a certain maximum size.

Another challenge will be for the AI to drive along the spline in an intelligent way. Cars controlled by the AI have to be able to avoid obstacles.

They also need to be able to drive cars created by users without knowing how the car behaves in terms of acceleration, max speed and handling.

1.3 Aims & Objectives

The main aims are:

- Implementing a sandbox racing game
- Create a racing track on top of splines

The main objectives are:

- Create car editor
- Create splines
- Generate track on splines
- Combine previous objectives into one game
- Implement racing game rules

2. Research

The YouTuber DokipenTech [1] created an unreal engine 4 tool to create a road from a spline as shown in figure 1. He made four video tutorials about how he has done it using blueprints.
In addition, an employee of Epic made a live training [2], which is still available on YouTube. In that training video they show a sophisticated way to create meshes on splines, also using unreal engine 4 blueprints. See figure 2 for a preview of his work.

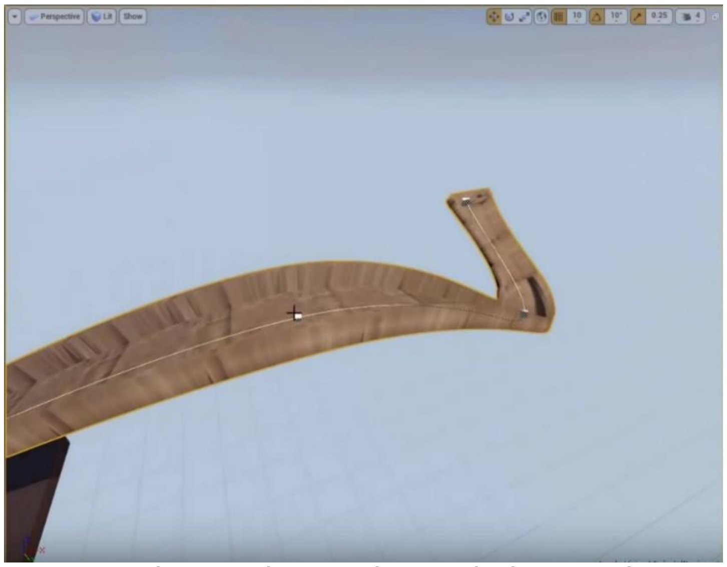


Figure 1: Road created on a spline with the unreal engine 4.

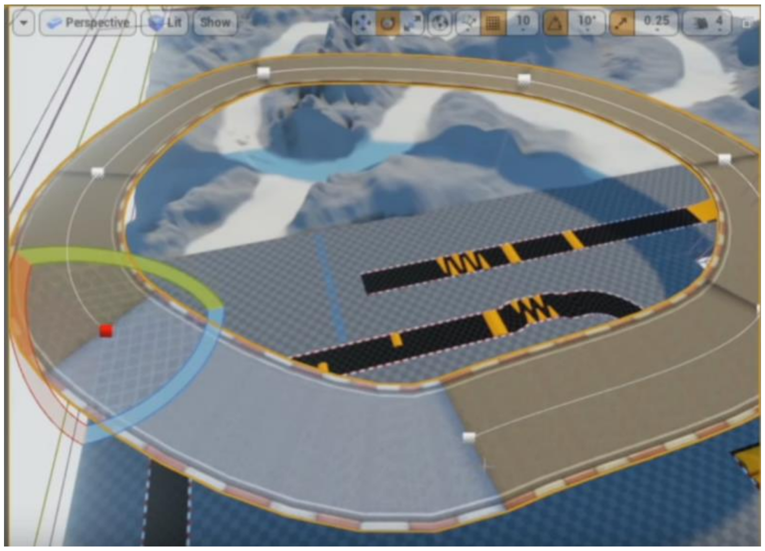


Figure 2: Spline road as created in a live training from Epic.

A sandbox vehicle editor has been done by several developers. Freejam LTD uses it in the game Robocraft [3] as seen in figure 3. Other games like Autocraft developed by Alientrap Games [4] also have the ability to create vehicles in a sandbox mode (see figure 4), though Autocraft does not use an external editor, just a mode switch in the current loaded level.
Another game that allows the player to create vehicles using blocks is Space Engineers by KEEN SWH LTD [5]. Space Engineers also does not switch into a separate editor and does not switch to a different mode to build as seen in figure 5.

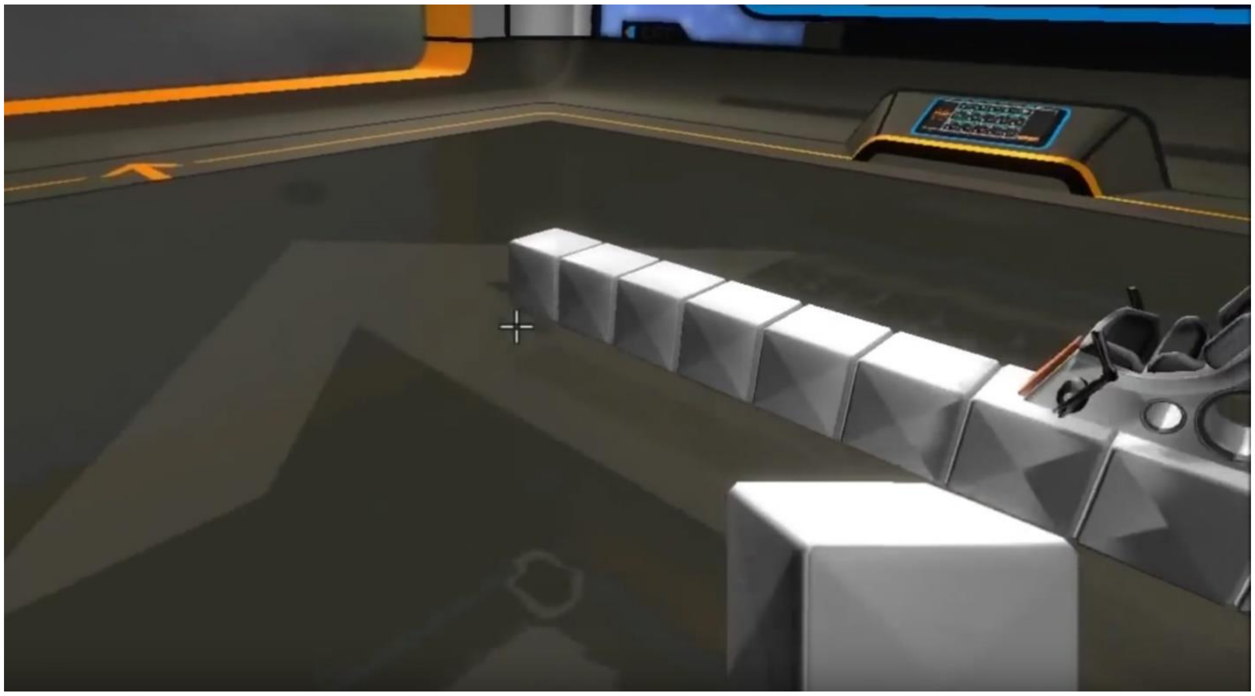


Figure 3: Robot builder used in the game Robocraft.

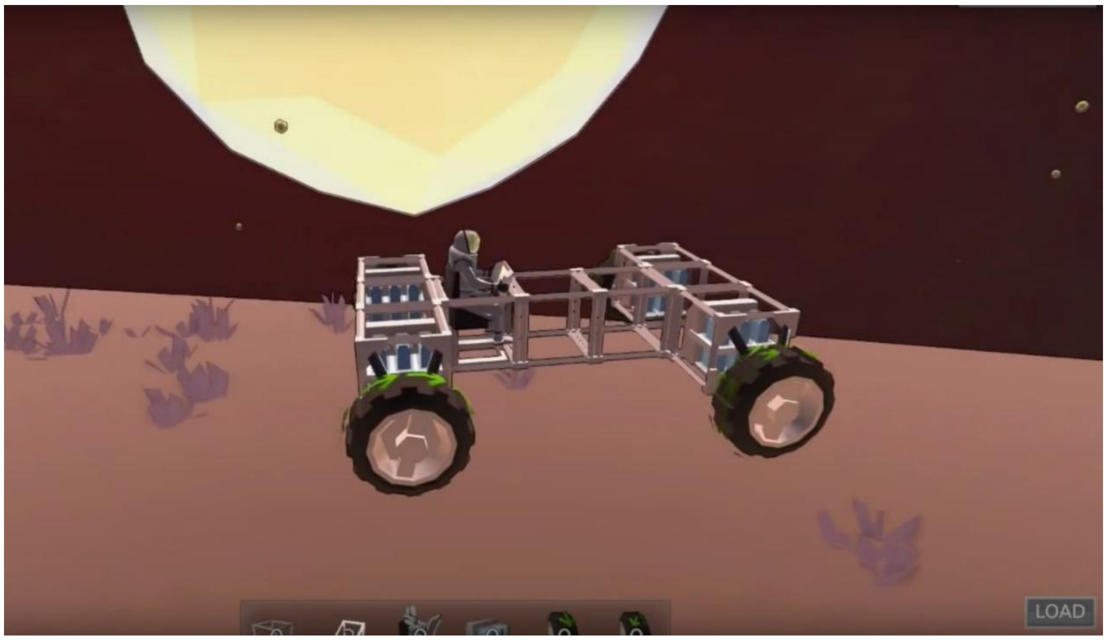
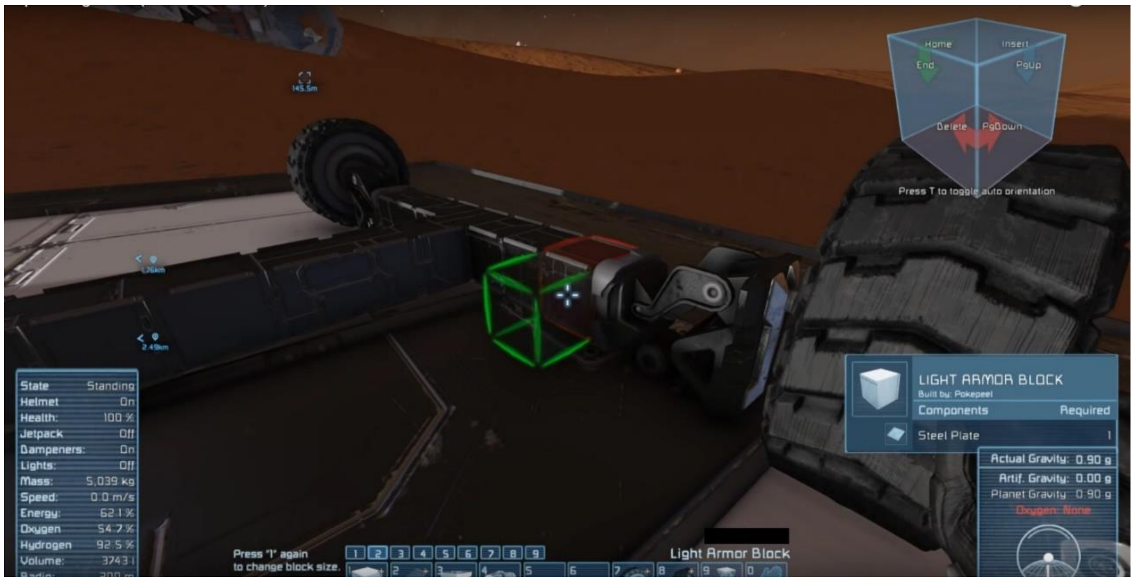


Figure 4: Building mode in the game Autocraft.



State Standing
Helmet On
Health: 100 %
Jetpack Off
Dampeners: On
Lights: Off
Mass: 5,039 kg
Speed: 0.0 m/s
Energy: 62.1 %
Oxygen 54.7 %
Hydrogen 92.5 %
Volume: 3743 l

LIGHT ARMOR BLOCK
Built by: Pokepeel
Components Required
Steel Plate 1

Actual Gravity: 0.90 g
Artif. Gravity: 0.00 g
Planet Gravity 0.90 g
Oxygen: None

Figure 5: Building a car in Space Engineers.

I could not find any implementation details of their editors. Therefore all information I have are from images and trying them out to get a feel of their functionality and controls.

3. Solutions

I will use the Unreal Engine 4 to implement the game, as it already provides helpful tools to expand upon. It already has built-in spline components and a sophisticated car controller to use as a base for the project.

To give players the ability to create their own cars, I will provide them with a separate editor mode as it is done in Robocraft. This allows me to constraint the cars size and make sure it is valid before it will be used to race. A separate editor makes also makes it easier to come back later and edit previously created cars.
There will be a virtual 3D-grid to place the blocks on and the editor will provide the player with a starting block. The player will only be able to place blocks attached on any other block and not free floating in the air. If they remove a block from the car, all blocks neighbouring the removed one will check if they still have a connection to the seat block. I will implement the check with a 3D version of the A* algorithm [6], which will find a way to the seat block if there is one. If there is not a route, it will notify the player by tinting the block red.

To create the spline roads, I will follow the tutorials mentioned before and extend upon it. The tutorials only show the basics to make it work, but there is more work needed to make it look “good” and to make sure that curves cannot be too steep. Furthermore, I decided to switch from blueprints to C++ as the extensions to the road generator will get very complex very fast if done in blueprints.

The AI will at first just use the spline as a path to drive on and will calculate the optimal speed by the steepness of curved, which it will detect through the angle of the tangent of a spline node point. They will drive along the spline using the steering algorithm, which also makes sure they avoid collisions.
Later in the development I plan to improve the AI so that it calculates a better path and better detect the optimal speeds also taking into consideration the cars specs.

4. Project Management

4.1 Task breakdown

I created tasks for the different parts of the game (see table 1). Everything to do with the track generator gets an ID starting with TG, car editor tasks have an ID starting with CE, the AI tasks begin with AI, asset tasks with AS and lastly the tasks which combine all the different parts into a game got IDs starting with GA. In addition, I gave each task a priority to indicate whether a feature is necessary (must have), will improve the game further (should have) or are entirely optional and might not be implemented due to the time needed to implement the game. In the task table, I also provided dependency to show which tasks have to be implemented in order to implement a certain other task. This ensures a smooth workflow, as I will not start tasks that cannot be completed at that point of work. Lastly, I provided a simple description of the task and what its implementation is supposed to achieve.

ID	Priority	Dependency	Description
TG001	Must Have		Generate the road mesh along the spline
TG002	Must Have	TG001	Apply road texture to road mesh
TG003	Optional	TG001	Make looping's possible
TG011	Must Have	TG001	Add attributes to spline points to control width, thickness, bank and element type
TG012	Must Have	TG011	Add a start line onto the track
TG013	Must Have	TG011	Add goal line onto the track
TG014	Should Have	TG011	On closed loop make start to also be goal
TG015	Should Have	TG011	Add booster to speed up car when they are on it
Completed up to here			
TG101	Optional	TG001	Randomly generate spline
TG102	Optional	TG101	Randomly change spline point attributes
CE001	Must Have		Level to build car in
CE002	Must Have	CE001	UI for editor
CE003	Must Have	CE001	Place start block which can't be removed
CE004	Must Have	CE001	Let the player place blocks on other blocks
CE005	Must Have	CE004	Let the player delete placed blocks
CE006	Must Have	CE005	Check for car validity
CE011	Must Have	CE001	Let player select block types from menu
CE012	Should Have	CE011	Add different weight chassis blocks
CE013	Must Have	CE011	Add wheels
CE014	Should Have	CE011	Add engines

CE015	Optional	CE011	Add fuel tanks
CE021	Should Have	CE011	Add colour picker
CE022	Should Have	CE021	Colour chassis blocks
CE031	Must Have	CE002	Allow naming of car
CE032	Must Have	CE006	Allow saving of car to a file
CE033	Must Have	CE032	Allow loading of car from a file
CE041	Optional	CE033	Test drive car
CE051	Optional	CE032	Bake car data to use less colliders and meshes
GA001	Must Have	TG;CE	Main Menu
GA002	Must Have	GA001	Car selection
GA003	Must Have	GA001	Level selection
GA011	Must Have	GA002;GA003	Load Level
GA012	Must Have	GA011	Spawn player
GA013	Must Have	GA012	Controls
GA014	Should Have	GA011;AI	Spawn AI
GA015	Must Have	GA013	Win condition
GA021	Must Have	GA011	UI
GA022	Must Have	GA021	Show speed
GA023	Should Have	GA021	Show lap/laps
GA024	Should Have	GA021;GA012;GA014	Show place
AI001	Should Have	GA	AI cars drive along spline
AI002	Optional	AI001	Avoid collisions with other cars
DC001	Must Have		Create task list
DC002	Must Have		Project proposal
DC003	Must Have	DC001;DC002	Progress report

DC011	Must Have	DC003;GA	Final report
AS001	Must Have		Road mesh and texture
AS002	Must Have	AS001	Start mesh and texture
AS003	Should Have	AS002	Goal mesh and texture
AS011	Must Have		Chassis block mesh and texture
AS012	Must Have		Wheel mesh and texture
AS013	Should Have		Engine block mesh and texture
AS014	Should Have		Fuel tank mesh and texture
AS021	Must Have		UI textures

Table 1: Task list with IDs, priority, task dependencies and descriptions.

4.2 Time Assessment

I estimated the time each task will take to be implement. For that, I created another table that shows the tasks category, ID and time estimate. The very last row is the sum of the time each task will need to be completed. The overall time estimate for all tasks amounts to **411** hours of work time, presumed there will not be any major problems or difficult to find errors, who increase the tasks time much.

Track Generator		**Car Editor**		**Game**		**AI**		**Assets**		**Documents**	
Task	**Time**	**Task**	**Time**	**Task**	**Time**	**Task**	**Time**	**Task**	**Time**	**Task**	**Time**
TG001	30,0 h	CE001	15,0 h	GA001	4,0 h	AI001	8,0 h	AS001	5,0 h	DC001	4,0 h
TG002	5,0 h	CE002	15,0 h	GA002	4,0 h	AI002	10,0 h	AS002	2,0 h	DC002	6,0 h
TG003	8,0 h	CE003	6,0 h	GA003	4,0 h			AS003	1,0 h	DC003	8,0 h
TG011	20,0 h	CE004	8,0 h	GA011	2,0 h			AS011	2,0 h	DC011	12,0 h
TG012	15,0 h	CE005	3,0 h	GA012	1,0 h			AS012	6,0 h		
TG013	10,0 h	CE006	20,0 h	GA013	3,0 h			AS013	3,0 h		
TG014	6,0 h	CE011	7,0 h	GA014	2,0 h			AS014	2,0 h		
TG015	6,0 h	CE012	2,0 h	GA015	1,0 h			AS021	6,0 h		
TG101	30,0 h	CE013	6,0 h	GA021	3,0 h						

TG102	10,0 h	CE014	2,0 h	GA022	1,0 h						
		CE015	2,0 h	GA023	1,0 h						
		CE021	16,0 h	GA024	2,0 h						
		CE022	3,0 h								
		CE031	1,0 h								
		CE032	8,0 h								
		CE033	8,0 h								
		CE041	10,0 h								
		CE051	36,0 h								
	140,0 h		168,0 h		28,0 h		18,0 h		27,0 h		30,0 h

Table 2: Time estimation to complete every individual task.

Since the car editor is the biggest feature of the game, it will also take the most time to complete. CE001 and CE002 have gotten a bigger amount of time than most tasks in the car editor category, since they provide the basic for the entire car editor and everything builds up on those. CE006, which is the car validity checker, has also gotten a large amount of time allocated, since it has to be very efficient as it might get executed several times per frame and should not increase the frame time by a noticeable amount. To implement and optimize the algorithm to the desired degree will take a long time, which is why the allocated time is set to 20 hours.
The largest time allocated to the car editor is the mesh and collider baking of the car, which will decrease computation and render times in the races considerably. It is also set to optional, since it might not be implemented in the time I have gotten to complete the game. It will implemented last, if there is time left, or after the project submission.

The second biggest category is the track generator. Again the first task has the longest time allocated to it, since it will provide the basis for each other task to build upon. TG101 and TG102 are also optional tasks, which might not be implemented, since they will need a lot of work and fine-tuning, but are not necessary for the game to function properly.

The basic implementation of the AI will probably not take longer than 18 hours, since they will only drive along the spline, which is already given and try to avoid other cars, which should not be much of a problem using the steering algorithm.

Assets will mostly be gathered from free sources or created by myself; they will be plain looking assets, as they are not necessary for the games functionality. They will be improved after the project submission.

The Game category will consist of the combination of all other categories and also provide menus, win conditions, detection of the games states and so on. Combining the previous parts should go rather quickly, since they already have everything necessary to load the correct data.

4.3 Risk Analysis

To prevent the loss of the project due to technical errors, I am using version control. I am using GitHub as a version control service (See Appendix A for the link). This allows me to keep a copy on a different server and it allows me to revert changes to the code that may cause errors in the game.

To make sure, the game will reach a certain point of completion in that it can actually be player, I will complete the task according to their priorities and dependencies. Task with the priority of optional will be done as the very last, since they are high risk and do not provide enough return to do them earlier.
The “Must Have”-priority tasks are the ones to provide a huge return and should therefore be completed at first.

Code optimization and AI improvements are not listed in the task list, since they will only be done if there is time left, after all other tasks have been completed, since especially the optimization may break a lot of code and is therefore very high risk.
Changes to the AI will change the entire game balance and experience and should be thoroughly tested, which will cost a lot of time.

5. References

[1] *Unreal engine 4 road tool in blueprints with Spline and SplineMesh - part 1 Spline components*, 2014 . YouTube

[2] *Using Splines & Spline components | live training | unreal engine*, 2014 . YouTube

[3] MIKECODEJAMMER and FREEJAM LTD, 2017. *ROBOCRAFT* [viewed 3 March 2017]. Available from: http://robocraftgame.com/

[4] ALIENTRAP GAMES, n.d. *Autocraft - early alpha access* [viewed 3 March 2017]. Available from: http://www.autocraftgame.com/

[5] KEEN SWH LTD, 2013. *Space engineers* [viewed 3 March 2017]. Available from: http://www.spaceengineersgame.com/

[6] Cui, X. and Shi, H., 2011. A*-based pathfinding in modern computer games. *International Journal of Computer Science and Network Security*, *11*(1), pp.125-130.

Appendix A

Link to the GitHub repository: https://github.com/NeoIzen/SploxRacers