Project Quacky Wheels: Development Schedule

By: Jade White, David Poeung, John Hall, and Marc DeSorcy

Intro:

This a rough schedule of what we plan on doing each iteration. This is by no means a complete comprehensive list of everything that needs to be done. Our development is an iterative process. The time and deadlines and order are also estimated. Our team also plans on using an online kanban board to keep track of these tasks and their progress.

Subtasks are denoted by indentation. Dependencies hold for every subtask below where it is listed. These tasks are ordered in the general order they should be done. If the task has no dependencies on other parts, they can be worked on concurrently.

If deadlines are on some milestone deadlines, it means they should be fully integrated and working by that point.

Tasks for Milestone 1:

These tasks are focused on choosing a extensible design and creating the basic skeleton of the application framework. Deliverables are the game design document, the feature list, a rough schedule, and the game application framework.

Task	Description	Who	Time	Deadline
Decide upon libraries and environments.	Decide what technology to use to implement the game engine. e.g. C++ 11? OpenGL 4.1? Multi Platform? etc.	Jade, David, John, Marc	< 1 day	Jan 20
Create Game Engine Skeleton	Design and implement the skeleton for the game engine. This affects every subsystem, so it is scheduled to be done first.	Jade, David, John, Marc	~ 6 days	Jan 25
(Sub-parts of creating Game Engine skeleton)	The following parts will be needed to be accounted for: - Game Loop (prob. monolithic) - Input - Rendering - Sound - UI - Physics - AI - Optional: script, content loading, animation	Jade, David, John, Marc	5 days	Jan 25

	Also, must decide on how game state will be kept, as well as how to share state.			
Design game engine skeleton	Design a class diagram of the general class layout of the game engine systems. Most importantly, outline the general design of the Entity, Component system - and how the Renderer, Input, and Sound will tie into it. This is not "Design the entire engine". Rather "Decide on a structure."	Jade, David, John, Marc	<= 1 day	Jan 25
Implement game engine skeleton	Implement the skeleton of the game engine, decided upon during design.	Jade, David, John, Marc	<= 5 days	Jan 25

Tasks for Milestone 2:

These tasks are focused on developing the engine to support basic capabilities for the "Red Brick" scene. This would be a test scene where we can drive our car. This iteration also includes several tasks to ready the engine for Milestone 3.

Task	Description	Who	Time	Deadline
Input	Create the Input class for accessing keyboard/ controller information. Integrate it into the game engine. Document the API for use with other components. *Possible dependency on Message Passing Interface.*	John, David	3 days	Jan 27
Controller IO	Read documentation on controller I/O Decide on data structures for storing and passing controller data Write algorithm for parsing controller input and returning data. Also, integrate it into the Input class.	John, David	3 days	Jan 27

Keyboard IO	Implement methods in the input class for keyboard input. Possibly these would allow us to use methods like bool GetKeyDown(Keycode code); In the Update() method. May consider an event-driven approach.	John, David	3 days	Jan. 30
Rendering	These are tasks associated with rendering. For this milestone, it is important we get "Red brick" rendering. Although personally, I'd like us to have textures at the end of Milestone 2.	Jade, David	12 days	Feb 5
Do research on a good rendering design.	Read documentation on OpenGL package. Code for possible relevant features (ex. particles, reflections, shadows, etc.). Decide on data structures to store graphical entities for rendering. (How to store meshes, materials, textures, shaders.). Consider how game engine components would tie in with these. (Personally, I would do something similar to Unity) Consider: World (Ambient light, Skybox) Objects (cars, powerups) Particles (billboard, sorting?) Special (particle effects, trails) You don't have to implement these features. Just keep them in mind. *Dependent on Data Loading and to a certain extent AI/Gameplay*	Jade, David (After IO)	1 day	Jan 27 (Should finish earlier)
Grey cube	Implement a renderer capable of displaying a standard "Grey Cube". The program should also be able to load objects.	Jade, David (After IO)		

	Write Initialization code Display Grey Cube		2 days 1 day	Jan 29 Jan 30
Red brick and meshes.	Push that a bit farther, a renderer capable of displaying a red brick car. Also arbitrary meshes. Red brick car Arbitrary meshes	Jade, David (After IO)	 1 day 2 days	Feb 2
Textured Mesh.	Push that even farther, and make the mesh capable of being textured. This will require a .obj importer, Implement the Mesh Source Component.	Jade, David (After IO)	2 days	Feb 4
Integrate with components	If you haven't been tying it in with the game engine skeleton this entire time, do so now.	Jade, David (After IO)	1 day	Feb 5
Physics	Design and implement initial physics system.	John, Marc	~3 wks	Feb 15
Research	Read documentation on PhysX Decide on relevant features to implement first. What kind of objects, which libraries. Decide on data structures to store objects under physics control These may be similar to rendering data structures. Also consider how these will tie in with the Component system/Engine. (Unity's system has a lot of hidden contracts, we can do better than that!)	John, Marc	1 day 1 day 1 day	Jan 27 Jan 28 Jan 29
Box Collisions	Implement code allowing us to collide two cubes in our test scene. One being the car, one being the floor. Be sure to tie this in with the game engine. You	John, Marc	3 days	Feb 3

	shouldn't need box rendering to be complete to get a start on this.			
Box - Mesh Collision	Implement code allowing us to collide a box with a mesh collider. This will be important for detecting the track.	John, Marc	2 days	Feb 5
PhysicsUpdate (FixedUpdate)	Add physics to the update pipeline in the engine (if not already)	John, Marc	1 day	Feb 8
Implement other colliders	Iteratively add other physics entities that would be useful and test/tune as necessary. **Springs**, and **Wheel** colliders are of special interest.	John, Marc	3 days	Feb 11
Raycasting	Add methods for ray casting, and be sure it's accessible to the Component API.	John, Marc	1 day	Feb 12
Scripting	These are tasks that are for making our work easier in the long run. E.g. a api for loading new levels, loading media, message passing etc. For loading art assets, maybe a easy-to-access API, like object Assets.Load(string filepathInAssets/).	Jade,	6 days	Feb 8
Research	Decide on formats for storing content levels, models, sound, etc. Decide how this should be stored/loaded Look for different ways to serialize public fields in C++ for Component Scripts. Don't think it's possible, so look into different workarounds that would allow us a similar freedom. E.g. a OnLoad() method. (Saving/Loading/(Updating?) possible tweakable values from text). *Dependent on beginnings of Rendering, Sound.*	Jade	2 days	Feb 10

Create basic classes	Create some standard classes that would accelerate later development. For instance, Vector3, Matrix4x4, Quaternion (If they don't already exist).	David	1 Day	Feb 7
Red Brick Scene	These are tasks with creating a test red brick rendering scene for the deliverable	Marc	3 Days	Feb 15e
Driving model	Implement a basic car using the physics system. Tweak until it drives in a nice manner.	John	1 Day	Feb 13
Test Track	Develop a test scene to drive the car on. This includes a bit of art if you have time. Model Track Texture Track	Jade	 2 Days 2 Days	Feb 14
Lap System (Time permitting. If not, Milestone 3)	Develop a simple spline, which will allow us to track the number of laps around the track each racer has done. Allow for multiple paths.	Jade	3 Days	Feb 15
Debug messages and Lines. (Time permitting. If not, Milestone 3)	Code some super simple code that allow us to display a debug log on screen. Also implement functions for displaying debug lines for help with vectors. Research Implementation	Jade	 2 Days 1 Day	Feb 15

Tasks for Milestone 3:

These tasks are for creating the game we envisioned on top of the engine. It also includes tasks for more engine features, like sound, that were not required for previous iterations.

Task	Description	Who	Time	Deadline
Scripting:	These are additional scripting tasks to make gameplay easier to manage.	Jade	4 days	Feb 19

Level Loading	Create a api that handles unloading/loading levels, which we could call from components. Stretch would be async. The engine should already be designed to allow this. Implement Engine Backend Implement API	Jade	 3 days 1 day	 Feb 18 Feb 19
Spline System	Implement a useable “waypoint” system. We will use this to implement the trails.	Jade	3 days (from Milestone 2)	Feb 22
Sound	These are tasks to do with implementing the sound system.	David	12 days	Feb 27
Research	Read documentation on sound package (SDK) Decide on sound file format Decide on which sounds to use and find files online (collision, missile, engine, etc.)	David	 1 day 3 days	 Feb 16 Feb 19
Implementation	Implement the audio entity components. Implement the low-level audio system classes.	David	4 days	Feb 23
Sounds in Game	Add some sound effects into the game where they fit. *Dependency on AI/Gameplay*	David	4 days	Feb 27
Gameplay	These are tasks regarding implementing the actual game logic.	Jade, John, Marc	4 days	Feb 27
Basic lap based gameplay	Implement code to detect a standard “game flow” (Countdown, Go, complete laps, display results). This can be very messy (i.e. non-reversable), the level loading system can simply destroy and recreate the scene.	Jade	4 days	Feb 27

Kart Driving	Using the red brick driving model, improve it to match a racing model found in our game design. Remember to leave hooks to make it multiplayer. For the most part, it has a Throttle, Brake, Drift state, Boost state, and so forth (test and tune).	John	~2 wks 2 days 2 days 1 day 1 day 1 day	Feb 23 Feb 16 Feb 18 Feb 19 Feb 22 Feb 23
AI	Using the same Kart driving model, create a simple AI to follow the track. Read up on behaviours and how to model them. Decide on opponent general strategy (How aggressive? What powerups? When to change tactics ex. ahead/behind?) Write code for AI entity Tune until the AI is challenging and fun Leave options for difficulty levels (stretch). *Dependent on Rendering and basic Physics*	Marc	~3 Days	Feb 19
Research	Look into techniques of racer AI.	Marc	1 Day	Feb 17
Racer AI	Implement racer AI, add a bit of jitter to make them more random.	Marc	2 Days	Feb 19
Tweaking (Ongoing)	Tune the variables until the driving feels natural.	Jade, David, John, Marc	∞	April 13
Powerups (Continuous through Milestone 4.)	Work towards implementing working powerups for the game. Implement one at a time, and be sure to test them. This is a huge task that will carry over into milestone 4.	John	4 days	Feb 29

	Speed boost powerup Jetpack powerup (upgrade version) Remaining powerups diverted to milestone 3		2 days 2 days	Feb 27 Feb 29
Rendering	Tasks related to rendering for creating the game prototype.	John	4 days	Mar 4
Trail Component	Create the mesh generator for the trails behind racers. Also create the scripting component for leaving trails.	John, Marc	2 days	Mar 2
UI System (Time permitting, see Milestone 4)	Start on the UI system from Milestone 4. It would be nice to have a working player UI.	John	2 days	Mar 4
Art (Time permitting, if not, then Milestone 4)	Create some good art for the game. Racer models, track modes, powerups etc.	Jade	4 days	Mar 27 or April 13

Milestone 4:

These tasks are centred around making the game "feature complete". This is finalizing the game, and adding polish (time permitting).

Task	Description	Who	Time	Deadline
UI	These tasks are for designing the main menu and various UI components.	John	2 wks	Mar 18
Research	Look into clever ways of designing a general and cool 3D UI system. Ideally, it should be able to handle mouse clicks, floating text in a 3D world, and displaying images preferably with masks. If time is running low, the UI system will likely have to be not general.	John	2 day	Mar 8
Implementation	Code up a UI system.	John	3 days	Mar 11

Main Menu/Player UI Design	Create a layout of how the UI should look. This on top of the concept art we have. What should be there, background art, structure of menus etc.	John	2 days	May 15
Main Menu & Options.	Create the UI elements for the main menu, and the options menu. Also a pause screen, time permitting.	John	2 days	Mar 17
Refined player UI	Refine the player UI, make it much more graphical. Tune until controls are intuitive and looks pleasing. Integrate into gameplay code. *Dependent on AI/Gameplay, Rendering*	John	1 days	Mar 18
Powerups	Implement Remaining Powerups based on remaining time Jump Walls Ignore Walls Remove Walls Solid Walls Mine Missile Teleport (stretch goal) Switch Places (stretch goal)	John	1 wk 1 day 1 day 1 day 1 day 1 day 1 day (as time permits)	Mar 28 Mar 21 Mar 22 Mar 23 Mar 24 Mar 25 Mar 28
Rendering	Iteratively add more complex graphics elements if they aren't done yet. Skyboxes Ambient Lighting Frustum Culling (Stretch) Shadows (Stretch) Motion Blur (Super Stretch)	Marc	~5 days	April 4
Skinned meshes (Stretch)	Look into implementing Skinned meshes. This would allow us to have animated characters. Likely to be put off priority.	Jade, Marc	3 days	April 11
Art	Create a more suitable race track, with some standard props. Ideally, make it so extra flare can be added later. Also	Jade		April 6

	design additional racers, and paint the assets. Basically add as much flare as you can in the time provided. Meshes Textures		 2 Days 2 Days	
Scripting	Tasks related to scripting. These would mostly be tweaking the current kart gameplay and implementation of other features. Like the scripting behind a better speedometer.	David	3 days	Mar 31
Multiplayer	Implement split screen multiplayer for the game.	David	3 days	April 3
Split Screen	Render the scene twice from the perspective of two different cameras. *Dependent on rendering.*	David	3 days	April 6
Rework scripting	Change the scripting and AI to accommodate two players in a race. This includes rendering an additional UI that tracks the second player's statistics. *Dependent on AI/Gameplay, Rendering, UI*	David	3 days	April 9
Implement second controller I/O	Rework or add additional methods to the I/O program to allow for input of a second controller that corresponds to a second racer. *Dependent on Input: Controller I/O*	David	3 days	April 13
Animation (Stretch)	Create a State-Machine based animation system. This would allow us to create a kart system with fluid animations more easily. Likely to be put off priority.	Jade		Mar 28 or April 13

	Research Low-Level implementation High level state machine		1 day 3 days 3 days	
Tween System (Stretch)	Create a Tweening system, this will allow us to animate arbitrary properties and values in a professional looking manner. Create evaluator Create various tween classes	Jade	 1 day 3 days	Mar 28 or April 13

Final Product Tasks:

These are tasks that will be worked on after Milestone 4. These are mostly polish tasks, as we should be feature complete. Milestone 4 will also serve as a "Catch up" period if we are behind.

Task	Description	Who	Time	Deadline
Bug fixing (Ongoing)	Try to remove as many memory leaks, crashes, and errors as possible.	Jade, David, John, Marc	3 wks	April 13
Polish (Ongoing)	Make the game feel as fun as possible, continue to tweak gameplay elements.	David	3 wks	April 13
Additional Art (Ongoing)	Continue adding refined artwork and graphics.	Jade	3 wks	April 13
Continue work from previous milestones. (Ongoing)	Complete stretch goals from previous milestones. As long as they don't destabilize too much of the game.(We are supposed to be "Feature Complete")	Jade, David, John, Marc	3 wks	April 13

Again, these are all estimated times, deadlines, and tasks. The result will vary as we continue our development.