



UNIVERSITY OF READING
DEPARTMENT OF COMPUTER SCIENCE

Extending a Platform Game in C/CC++: User Experience Enhancement

JASON JAY DOOKARUN

WORD COUNT: 2225

(EXCLUDING DIAGRAMS AND CODE SNIPPETS)

PAGE COUNT: 25

MODULE CODE: CS1PR16
ASSIGNMENT REPORT TITLE: PROGRAMMING PROJECT
STUDENT NUMBER: 26017434
TIME SPENT: 50 HOURS
APRIL 21, 2020

Summary

The following document consists of an in-depth systematic examination of elements developed by I, Jason Jay Dookarun, to an existing C/C++ program developed by Parallel Realities. The given program is a platform game that employs the SDL2 library for the demonstration of graphics. A feature has been composed, modified and extended, integrated within the provided game, with the support provided by the skeleton code, authorised by **Dr Julian Krunkel** of the Department of Computer Science at the University of Reading. The aforementioned will further elaborate on the procedures undertaken to develop the feature(s) including design illustrations, methods of implementation and development process.

Declaration

I, **Jason Jay Dookarun**, of the Department of Computer Science at the University of Reading, confirms that all the sentences, figures, tables, equations, code snippets, artwork and illustrations in this report are original and have not been taken from any other person's work, except where the works of others have been explicitly acknowledged, quoted and referenced. I understand that if failing to do so will be considered as a case of plagiarism. Plagiarism is a form of academic misconduct and will be penalised accordingly.

Jason Jay Dookarun
April 21, 2020

Contents

Summary	i
Declaration	i
List of Figures	ii
1 Introduction	iii
1.1 Project Goals	iii
1.2 Game Features	iii
1.3 Programming Style	iv
2 Design	iv
2.1 Feature Design	iv
2.2 Gameplay after Modification	iv
2.3 Illustrative Design	iv
2.3.1 Flowchart	v
2.3.2 UML Diagram	v
3 Implementation and Development	vi
3.1 Control Panel Modifications	vii
3.2 Introductory "Splash" Screen Creation	viii
3.3 Outro Screen Creation	ix
3.4 In-Game Support Panel	x
4 Conclusion	xi
5 Appendix A: Full Flowchart	xiii
6 Appendix B: Code Changes	xiii
7 Appendix C: Player.c After Modifications	xvii
8 Appendix D: Homescreen.c After Modifications	xviii
9 Appendix E: Outro.c After Modifications	xxii

List of Figures

1 Pete's Pizza Party: Pre-Extension	iii
2 Introduction Flow Chart	v
3 UML after Modifications	vi
4 Introduction Screen	vii
5 In-Game Support	x
6 Flowchart	xiii

1 Introduction

The provided program employs an SDL2 library to illustrate graphical components, coded in C/C++. This program has been developed and created by Parallel Realities titled "Pete's Pizza Party 6". The present project has been produced using the SDL2 library setup. The current game enables a user to control a sprite to collect all objects illustrated on the screen appropriately before the termination of the game.

1.1 Project Goals

This project aims to develop an extension to the existing piece and further describe the choice made and implementation cycle applied. Authorised by the academic staff members at the University of Reading, I was provided with a skeleton code containing sections of the existing platform game, namely "Pete's Pizza Party 6".

1.2 Game Features

The aim of the game entails accumulating all objects positioned across disparate seconds of the map. These objects would be portrayed as pizza slices. As the user assembled every pizza slice, the HUD, located on the right-hand side of the user's screen, would increment, accordingly. Once the entity collected all the elements, the game would be terminated.

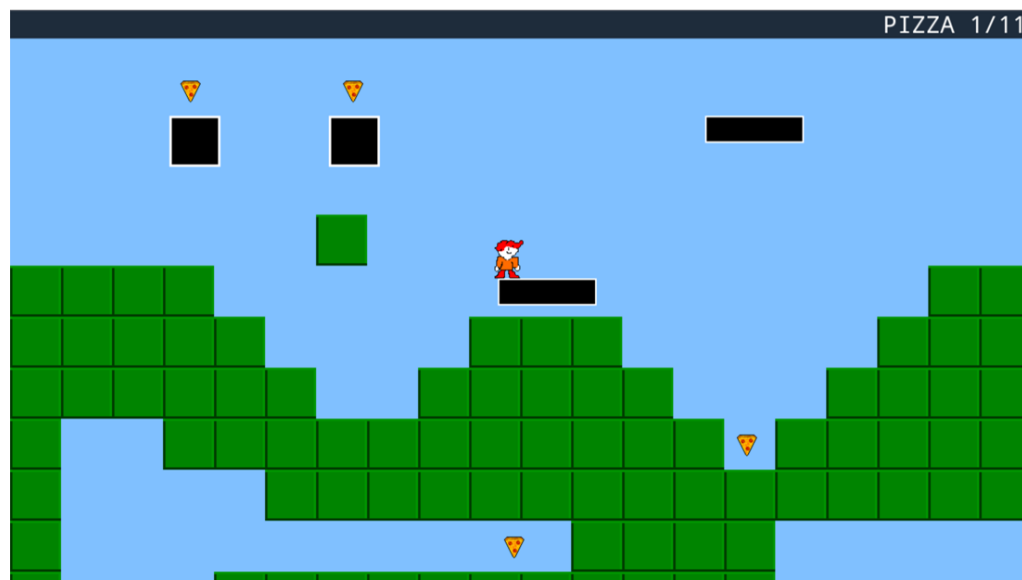


Figure 1: Pete's Pizza Party: Pre-Extension

After running several instances, I was able to identify numerous components that could've been adjusted or implemented to provide the user with enhanced user experience, namely universal control commands, a support panel and an introduction sequence to allow the user to comprehend the product.

Currently, the product implements any player movement the use of the letters A, D, I and SPACE, where A represents a movement to the left, D represents a movement to the right, I represents jump, and SPACE represents a reset functionality. This could be potentially identified as a difficulty for the user, as a universal control panel is not implemented. Moreover, no support is provided to the

user for one to understand what the controls may be. This sector can, as a result, be identified as a weakness, allowing such an extension to be provided.

Moreover, once the program is executed, the user is immediately provided with the main program. As a result, the user is not provided with a formal welcome to the game and intricately links to the avoidance of positive user experience. Furthermore, once the user does complete collecting all objects displayed on the map, the user is not provided with an option to play again, but instead is forced to view the game terminated. Both of the raised weaknesses interlink with the subject of user experience, as both consequently do not fulfil positive UX (User Experience) for the user. This, as a consequence, provided me with a segment to converge upon, to transform and intensify the product for future clientele.

1.3 Programming Style

To achieve my modifications, I originated by performing a bottom-up approach. [3] A bottom-up method fulfils modifications commencing with the evaluation of existing components within the program, followed by constructing more complex features. This can be considered as an asset as it reuses existing methods, eradicating the need for additional examination. I will be programming in C throughout the course of the project to inject my modifications coherently.

2 Design

To coherently comprehend the flow of a project, it is important to implement a design structure, whereby one can visualise before building or modifying any designated section. Moreover, to ensure that the project is understandable, levels of abstraction can be implemented at levels like design, allowing communication to take place between teams.

2.1 Feature Design

Following my brief analysis of the program before development, I believe that it would be useful to focus on changes correlating to one's perception of the user's experience. This would include an introductory screen to welcome the user, implementing an in-game support mechanism, to further support the user during live game-play. Moreover, modifications are to be made to the controls the player must follow, to adhere to a universal control method. Finally, a closing progression would be performed to allow the user to either: play the game again or exit the game appropriately, as an alternative to a forceful closure by the machine.

2.2 Gameplay after Modification

Once these modifications have been added, the game would commence from the main introductory screen as an alternative to the main gameplay screen. This would act as a welcome panel for the user, to comprehend the game as well as controls before playing. Once the user presses SPACE, the game will initiate as usual.

New controls are to be executed to follow a WASD or Up Down Left Right universal control setup. Once the game is finished, the user will then be shown an outro screen, presenting them with a possibility to either restart the game or exit the game, as an alternative to a force stop. New features are to be implemented, whereby the user can exit the game at any stage as well as an option to view controls via a key command.

2.3 Illustrative Design

To symbolise my concepts appropriately, I selected to further complete research in such fields. This would allow me to comprehend where to insert my modifications to positively impact both the

project, as well as the client. Diverse techniques can be used to learn where changes should be added, including methods like UML diagrams and flowcharts.

2.3.1 Flowchart

A flowchart is defined as "a diagram that depicts a process, system or computer algorithm." A flowchart can be utilised in a countless number of ways, as mentioned above. Similar to methods depicted both in UML diagrams and pseudocode, a flowchart illustrates a physical construct on how one may define their "algorithm". To comprehend how a flowchart functions, a universal keyset it used, for future users to understand its meaning. For instance, in the format of flowcharts, diamonds often refer to a decision that one must take. Such structures can be of effective use, as it allows one to experience the flow of a project in a digital and directed manner.

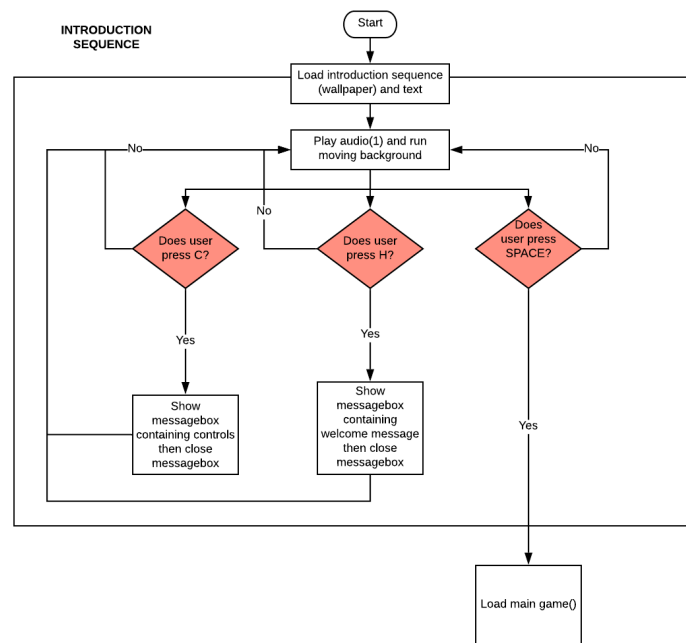


Figure 2: Introduction Flow Chart

As illustrated above, the flowchart implements my modifications that are to be executed. This solely focuses on the true nature of the introductory page that will be created. As pointed, numerous sections are given to the user through commands. These examples include pre-set reactions by the machine to key presses. This permits the formation of fluidity to the system, including a process of adhering to the user's experience.

2.3.2 UML Diagram

UML diagrams can be utilised as a form of representation, allowing a way of "visualizing a software program using a collection of diagrams." [8] A UML diagram can be useful by illustrating how objects interlink with one another. Likewise, in such scenarios, this enables me to learn the accurate position to infuse my advancements and modifications.

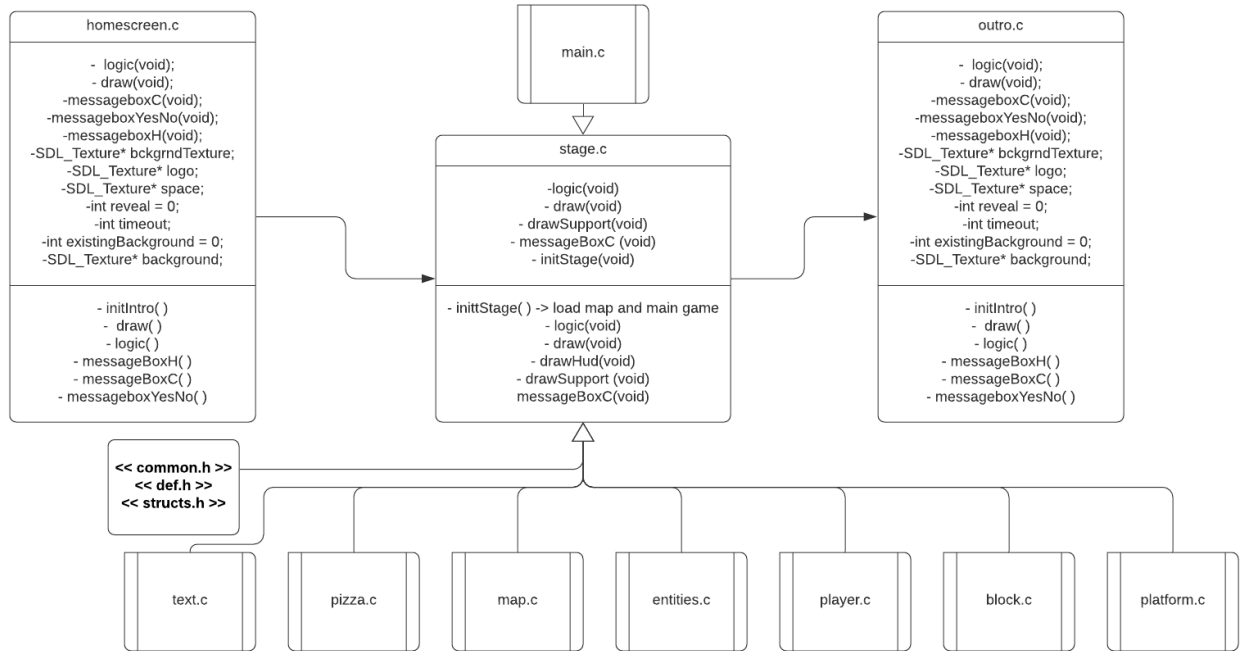


Figure 3: UML after Modifications

As illustrated in Figure 3, the new modifications would effectively implement a new introduction sequence and an outro sequence. Effectively, all the sectors in the main section are inherited from alternative sections as illustrated. On the other hand, the outro sequence and intro sequence will inherit similar features as one another.

3 Implementation and Development

As discussed in the design section, my modifications are to be specifically tailored to the user's experience. This includes the addition of an introductory panel for the user to be welcomed by, a modification in the controls sector by adding a universal control set, and an outro sector, providing the user with an opportunity to exit the game appropriately or replay.

To effectively implement my modifications, I began by applying the bottom-up approach as my principal methodology. This enabled me to modify existing sections, ensuring that both the program would resume working and certain sectors could be efficiently reused, thus reducing further testing. As a result, my initial segment of progress converged upon the control panel as this was a current section implementing minor modifications.

An immediate modification that was made to the game was the renaming on the program from "Pete's Pizza Party 6" to "Pete's Pizza Hunt". This provided a unique name to match the game accordingly. Graphics for both introductory panels and outros were designed through the use of Adobe PhotoShop CC.



Figure 4: Introduction Screen

3.1 Control Panel Modifications

To efficiently implement the new universal control methods, it was crucial to distinguish the present methods that were being employed in the program. Firstly, I started by examining the existing `player.c` file confined in the game, pre-development. This contained commands that the client was to be applying throughout the course of the game, before modifications to move the sprite.

```

1 void doPlayer(void)
2     if (app.keyboard[SDL_SCANCODE_A]){
3         player->dx = -PLAYER_MOVE_SPEED;
4         player->texture = pete[1];

```

Listing 1: `player.c` Initial Code

As illustrated by listing 1, the existing methods contained within the program illustrated the use of the key A and the key D, as shown by lines 5 and 8. Once these keys were to be pressed, the system would react in a designated method, i.e. move left or move right. This, as a result, provided me with an appropriate platform to implement changes.

```

1 if (app.keyboard[SDL_SCANCODE_LEFT])
2     {
3         player->dx = -PLAYER_MOVE_SPEED;
4         player->texture = pete[1];
5     }

```

Listing 2: `player.c` Modifications

Following the existing methods, I was able to apply the appropriate modifications by changing the scan-code to the appropriate key. This permitted the user to be provided with 2 manners of controlling the sprite, via WASD control method or the use of the Up Down Left and Right key prompts, accordingly (see Appendix B). This method was of extreme use as it allowed me to develop further configurations to benefit the player, later discussed. To ensure that testing was performed

accordingly, the game was executed over numerous cycles to ensure performance was maintained accordingly.

3.2 Introductory "Splash" Screen Creation

Once the modified controls were completed, I commenced working upon the introduction sequence. To understand how one may create this, I referenced the work and tutorial developed by Parallel Realities. This would allow me to understand what stages were required. This section, similar to stage.c required 3 concepts, logic(void), draw(void) and init() sequence. Similar to initstage(), initTitle() could be formulated identically to annul complications. With regards to the draw, the sector would implement the creation of the main screen as a platform to further build upon. As illustrated, lines 3-6 generate the panel for display, followed by line 7 formulating a render.

```

1 // implements the drawing of the wallpaper onto the screen
2 for (x = existingBackground; x < SCREEN_WIDTH; x += SCREEN_WIDTH){
3     screen.x = x;
4     screen.y = 0;
5     screen.w = SCREEN_WIDTH;
6     screen.h = SCREEN_HEIGHT;
7     SDL_RenderCopy(app.renderer, background, NULL, &screen);
8 }

```

Listing 3: homescreen.c background setup

Once this section successfully executed, further modifications were applied such as a control mechanism to take the user from one screen to another. The initial process involved designing/writing a message on the existing platform as shown below:

```

1 loadText(logo, &r, (SCREEN_WIDTH / 2) - (r.w / 2), 250);
2 // projected onto wallpaper as a visible instruction
3 drawText(SCREEN_WIDTH / 2, 450, 255, 255, 255, TEXT_CENTER, "PRESS SPACE TO PLAY!");
4 drawText(SCREEN_WIDTH / 2, 500, 255, 255, 255, TEXT_CENTER, "PRESS C FOR CONTROLS");
5 drawText(SCREEN_WIDTH / 2, 550, 255, 255, 255, TEXT_CENTER, "PRESS H FOR HELP");

```

Listing 4: homescreen.c command prompts

As demonstrated by the codes, the system requires the user to select a set of control to respond. Similar to the design section, such could be achieved by providing the user with a message box, and thus was implemented.

```

1 void messageboxC(void) {
2     // referenced from https://wiki.libsdl.org/SDL_ShowMessageBox#Version
3     const SDL_MessageBoxButtonData buttons[] = {
4         { SDL_MESSAGEBOX_BUTTON_RETURNKEY_DEFAULT, 0, "OK" },
5     };
6     const SDL_MessageBoxData messageboxdata = {
7         SDL_MESSAGEBOX_INFORMATION, NULL, "Controls", "Left: LEFT Key or A, Right : RIGHT Key,
8             Jump : SPACE or W, Reset : R, Quit : Q.", SDL_arraysize(buttons), buttons };
9     int buttonid;
10    if (SDL_ShowMessageBox(&messageboxdata, &buttonid) < 0) {
11        SDL_Log("error displaying message box");
12        return 1;
13    }
14    if (buttonid == 0) {
15        return;
16    }

```

16 }

Listing 5: homescreen.c message boxes

3.3 Outro Screen Creation

Similarly, by using methods applied within the introductory section, I developed an identical function to take place as an outro. This would specifically function in a manner whereby an alternative display will be shown to the user, with a new background soundtrack. This would be followed by a new command prompt allowing the user to 1) restart the game or 2) exit appropriately. Lines 4-5 initiate the processing of logic() and draw() prior to 7-8 loading content. Lines 11-12 initiate the soundtrack and play the appropriate audio.

```

1 // initiates the loading of the graphics and plays the background audio,
2 void initTitle(void)
3 {
4     app.delegate.logic = logic;
5     app.delegate.draw = draw;
6     // this allows the background wallpaper to be uploaded and displayed for the user.
7     background = loadTexture("gfx/backgroundNew.png");
8     logo = loadTexture("gfx/logo.png");
9     // the music score utilised in the introduction package is an adaptation of Stu
10    // Phillips' Knight Rider Theme, covered by
11    // Jason Jay Dookarun and recorded.
12    loadMusic("music/kr.mp3");
13    playMusic(1);
14 }
```

Listing 6: homescreen.c initTitle() sequence

```

1 void initOutro(void)
2 {
3     Mix_HaltChannel(-1);
4     app.delegate.logic = logic;
5     app.delegate.draw = draw;
6     background = loadTexture("gfx/outro.png");
7     logo = loadTexture("gfx/missionaccomplished.png");
8     // Music from : https://www.bensound.com
9     loadMusic("music/bensound-erf.mp3");
10    playMusic(1);
11 }
```

Listing 7: outro.c initOutro() sequence

As illustrated above, minor modifications were applied to both sections to ensure that both sections were to react in an appropriate manner, accordingly. Modifications further involved a change in procedures that followed following a key press as illustrated below:

```

1 // restarts the game by transferring the client to the introductory splash screen
2 if (app.keyboard[SDL_SCANCODE_P]) {
3     initTitle();
4 }
5 // similar to homescreen.c, allows the user to meet the exit message box to confirm
6 // their decision.
7 if (app.keyboard[SDL_SCANCODE_Q]) {
```

```

7     messageboxYesNo();
8 }
9 }

```

Listing 8: outro.c Key Press Process

3.4 In-Game Support Panel

As previously explained, the control mechanisms were developed accordingly through the use of keypress prompts. To ensure the user understood the function of the game during live gameplay, existing keypress commands were injected. This would permit the user to regularly review any controls as well as exit the game if needed through an exit panel. This was processed and completed by injecting the controls message box into stage.c, (view listing 5). As illustrated below, the exit mechanism was developed via a message box setup. As demonstrated, lines 2-4 initiate a button setup, allowing the user to click accordingly. The data would then be recorded (via line 5) prior to providing a set reaction, as shown by lines 8-15.

```

1 void messageboxYesNo(void) {
2     const SDL_MessageBoxButtonData buttons[] = {
3         { SDL_MESSAGEBOX_BUTTON_ESCAPEKEY_DEFAULT, 1, "Yes" },
4         { SDL_MESSAGEBOX_BUTTON_RETURNKEY_DEFAULT, 2, "No" },,};
5     const SDL_MessageBoxData messageboxdata = {
6         SDL_MESSAGEBOX_INFORMATION, NULL, "Exit Game", "Do you wish to exit the
           game?", SDL_arraysize(buttons), buttons };
7     int buttonid;
8     if (SDL_ShowMessageBox(&messageboxdata, &buttonid) < 0) {
9         SDL_Log("error displaying message box");
10        return 1;}
11    if (buttonid == 1) {
12        exit(0);}
13    else {
14        return; }
15 }

```

Listing 9: Exit Message Box

As a result, the following would be presented to the user when a keypress command was to be activated during live gameplay.



Figure 5: In-Game Support

4 Conclusion

In conclusion, I believe that my modifications have improved one's perception of the product by applying new methods to improve engagement. By applying sound effects to the introduction sequence and text animations, it allowed for a vaster audience to be interested in the product.

Developing a significant program like the one provided allowed me to understand from a different viewpoint of a customer what features affect one's perception, as well as methods of implementation, through research and further learning. Likewise, this project allowed me to apply different fields of knowledge such as my studies of Software Engineering as well as Programming at the University of Reading. Moreover, given the time frame provided by academic members, I was able to effectively understand the importance of time throughout the course of this project. This project allowed me to utilise La-Tex as a production method to develop my report, and I believe that this would be a skill that I would implement regularly to ensure professionalism is maintained throughout my future documentations.

If provided further time, I would try to implement another level to the program and change the constraint from the number of pizzas to collect to a time constraint. This would effectively interlink into the creation of a high score leader-board to form a further competition between players.

If I was able to re-complete the project, I would aim to focus more time towards the start of the project instead of delaying modifications I added in. This would have resulted in me in having more time to develop further features.

References

- [1] Parallel Realities, *2D Platformer Tutorial*
<https://www.parallelrealities.co.uk/tutorials/ppp/ppp1.php>
- [2] Parallel Realities, *2D Shoot 'Em Up Tutorial*
<https://www.parallelrealities.co.uk/tutorials/shooter/shooter15.php>
- [3] Bottom-Up Programming, *Bottom-Up Programming*
<https://bit.ly/2XE4JNs>
- [4] Science Direct, *Pseudocode*
<https://www.sciencedirect.com/topics/engineering/pseudocode>
- [5] SDL Wiki 2.0, *SDL Message Box*
https://wiki.libsdl.org/SDL_ShowMessageBox
- [6] SDL Wiki 2.0, *SDL Simple Message Box*
https://wiki.libsdl.org/SDL_ShowSimpleMessageBox
- [7] Don Norman and Jakob Nielsen, *The Definition of User Experience*
<https://www.nngroup.com/articles/definition-user-experience>
- [8] SmartDraw, *UML Diagram*
<https://www.smartdraw.com/uml-diagram/>
- [9] Lucid Chart, *What is a Flowchart?*
<https://www.lucidchart.com/pages/what-is-a-flowchart-tutorial>
- [10] Thorben Janssen, Stackify, *What is Abstraction?*
<https://stackify.com/oop-concept-abstraction/>

5 Appendix A: Full Flowchart

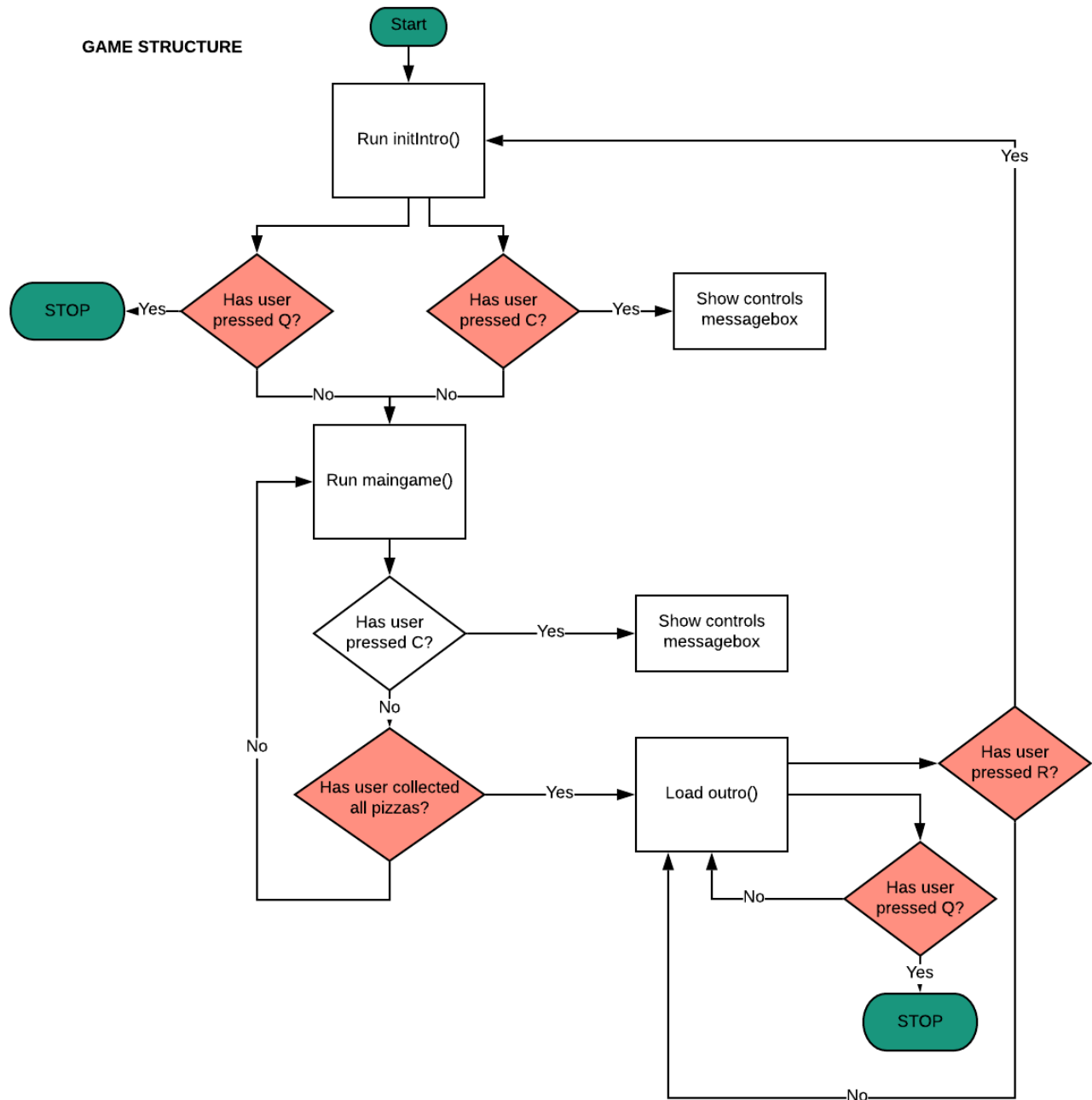


Figure 6: Flowchart

6 Appendix B: Code Changes

Code Changes: <https://csgitlab.reading.ac.uk/qt017434/CS1PR-Project>

Source	master
--------	--------

...



Target	4783d520f27febc8180925949e5c6a37b3b2e471
--------	--

Commits (31)
<p>Delete .gitignore (/qt017434/CS1PR-Project/commit/ed41f2c86327e352e02b91ae29c7314a9958ccb0) · ed41f2c8 Jason Dookarun (/qt017434) committed a month ago</p>
<p>initial commit (/qt017434/CS1PR-Project/commit/436435a4e4145e52aab9d2464412ab888abbdbd6) · 436435a4 qt017434 (mailto:jj.dookarun@student.reading.ac.uk) committed 3 weeks ago</p>
<p>implementation of up down left right as alternative to wasd (/qt017434/CS1PR-Project/commit/d082455cde4299a4560807e4725eb575f5a147df) · d082455c Jason Dookarun (/qt017434) committed 3 weeks ago</p>
<p>Implementation of quit command for user to quit game as alternative to Alt F4. (/qt017434/CS1PR-Project/commit/7f22ec1ed90a484ab7c57e2c0a6a1e44fd60faea) · 7f22ec1e Jason Dookarun (/qt017434) committed 2 weeks ago</p>
<p>application of new timer HUD, new module to be created (/qt017434/CS1PR-Project/commit/9f8317bdc55c1be3bcedbe34bc82ebec706343ac) · 9f8317bd Jason Dookarun (/qt017434) committed a week ago</p>
<p>addition of exit messagebox, requires completion (if YES is clicked then close) or apply timer (/qt017434/CS1PR-Project/commit/5e0f0c3a0a6e82dfc05953fcf3ca086e4620ce46) · 5e0f0c3a Jason Dookarun (/qt017434) committed a week ago</p>
<p>temporary changes to exit method of game, and completion of change 1 of the to do list (/qt017434/CS1PR-Project/commit/29025cabdd0f6e315195fb21511c3cd51c5ceb1f) · 29025cab Jason Dookarun (/qt017434) committed a week ago</p>
<p>Update: New homescreen has been created in order to provide a main page for the... (/qt017434/CS1PR-Project/commit/436b8a4ca43943816d8cf3e7808d9ba223229606) · 436b8a4c Jason Dookarun (/qt017434) committed a week ago</p>
<p>Future function added for logo and text implant (/qt017434/CS1PR-Project/commit/eebd1c336bd67c57701d7cbac68f0283dbe800e7) · eebd1c33 Jason Dookarun (/qt017434) committed a week ago</p>
<p>optimisation of map 1 to have a flat colour, and development of map 2 (/qt017434/CS1PR-Project/commit/b1d347d3ed69dd51b28c354ce8d9387e53f168cc) · b1d347d3</p>

Jason Dookarun (/qt017434) committed 6 days ago
new background for main splash screen added and bug fix from previous commit has... (/qt017434/CS1PR-Project/commit/dce7c9994fdc65f4643895358e2bb82be23f3434) · dce7c999 Jason Dookarun (/qt017434) committed 6 days ago
minor bug fixes. Implementation of Map 2 required next. (/qt017434/CS1PR-Project/commit/ac62b53d9931a984acb76e81c1f80873487a34a6) · ac62b53d Jason Dookarun (/qt017434) committed 6 days ago
minor changes to map to experimenet extensions (/qt017434/CS1PR-Project/commit/41a05f163178b6651c3e1130b94e1f3f1f7c879d) · 41a05f16 Jason Dookarun (/qt017434) committed 6 days ago
homescreen.c has been restructured, and implementation of help command has been... (/qt017434/CS1PR-Project/commit/f95110d8c786dee76c4ef19435984aed22891877) · f95110d8 Jason Dookarun (/qt017434) committed 6 days ago
updated to do list, next task is to create a link between L1 and L2 (/qt017434/CS1PR-Project/commit/cd6a209ef21f28ccf9400819670b20c4e5ec8d08) · cd6a209e Jason Dookarun (/qt017434) committed 6 days ago
minor bug fix with file locations. (/qt017434/CS1PR-Project/commit/74dff74c5c067bc9a8c295beda0433e7e2a97896) · 74dff74c Jason Dookarun (/qt017434) committed 6 days ago
minor changes to variable names to ensure it is cleaned up. (/qt017434/CS1PR-Project/commit/116ad30340461aae6301daedf5a3e8792b30a493) · 116ad303 Jason Dookarun (/qt017434) committed 6 days ago
updated to do list in order to keep track of progression (/qt017434/CS1PR-Project/commit/1a523c9a98c42823714015b8f4720a6d56632ee7) · 1a523c9a Jason Dookarun (/qt017434) committed 6 days ago
Implementation of messagebox.yesno applied to exit command for user experience. (/qt017434/CS1PR-Project/commit/130d1baff90f482fe34166174edd17d87cf52028) · 130d1baf Jason Dookarun (/qt017434) committed 6 days ago
updated to do list (/qt017434/CS1PR-Project/commit/9233bbb7646ced6e2b53c04225d08ec626e2b3f7) · 9233bbb7 Jason Dookarun (/qt017434) committed 6 days ago
level 2 successfully loads into game but does *not* load pizza. Target for next fix. (/qt017434/CS1PR-Project/commit/571504f8ba6c18b09fc2eab6f69d09ae60e87b1a) · 571504f8 Jason Dookarun (/qt017434) committed 6 days ago
updated checklist for future reference. (/qt017434/CS1PR-Project/commit/bd64dd6b3fc2e7335f23ff60522349e84580149b) · bd64dd6b Jason Dookarun (/qt017434) committed 6 days ago
audio fix complete. (/qt017434/CS1PR-

Project/commit/d4f56d7538dc8a424fb3f6d335ee61281a576514) · d4f56d75 Jason Dookarun (/qt017434) committed 6 days ago
added audio to intro and main gameplay. (/qt017434/CS1PR-Project/commit/c849a1c785b6bd3747cbf31f6300a21496c7a64a) · c849a1c7 Jason Dookarun (/qt017434) committed 6 days ago
audio successfully working for both intro and main gameplay (/qt017434/CS1PR-Project/commit/1fd351802fd195470e18799d0350a0937ce0b9d5) · 1fd35180 Jason Dookarun (/qt017434) committed 6 days ago
updated checklist, (/qt017434/CS1PR-Project/commit/0f18e6392ffa8c2af745fccd7ff1345a023ff1bd) · 0f18e639 Jason Dookarun (/qt017434) committed 6 days ago
enhanced complexity for buttons with an additional OK. (/qt017434/CS1PR-Project/commit/8480e355c37dec6ee62d81317f2466c73689887) · 8480e355 Jason Dookarun (/qt017434) committed 5 days ago
outro sequence implemented once user completes game. (/qt017434/CS1PR-Project/commit/4be2f776602261527115c2c6ae246bf586cc5c61) · 4be2f776 Jason Dookarun (/qt017434) committed 2 days ago
creation of latex report, to be developed by overleaf.com (/qt017434/CS1PR-Project/commit/f42af05cd9a42d8a42c8e754df6dfbb7dec02b9b) · f42af05c Jason Dookarun (/qt017434) committed 2 days ago
Outro applied and support panel on top (/qt017434/CS1PR-Project/commit/0f9dd6eba7c7bb667fe832c9e4f0d0ba3a58361b) · 0f9dd6eb Jason Dookarun (/qt017434) committed 2 days ago
applying outro track with copyright applications and commented sections. (/qt017434/CS1PR-Project/commit/2bc2b79daada478d355dc805a2d2860e0af76809) · 2bc2b79d Jason Dookarun (/qt017434) committed a day ago

Showing 855 changed files ▼ with **3508 additions** and **0 deletions**

▼  .gitignore
1 + a.out
2 + *.o
3 + main
4 + /project-visual-studio/SpringProjectx/Debug/SpringProject.tlog
5 + /project-visual-studio/SpringProjectx/.vs/SpringProject/v16
6 + /.vs/SpringProject/v16
7 + /.vs
8 + /project-visual-studio/SpringProjectx/sound
9 + /project-visual-studio/SpringProjectx/x64/Debug/SpringProject.tlog
▼  lecture-examples/autumn/10/2d-array-ragged.c 0 → 100644

7 Appendix C: Player.c After Modifications

```

1
2 void initPlayer(void)
3 {
4     player = malloc(sizeof(Entity));
5     memset(player, 0, sizeof(Entity));
6     stage.entityTail->next = player;
7     stage.entityTail = player;
8
9     player->health = 1;
10
11     pete[0] = loadTexture("gfx/pete01.png");
12     pete[1] = loadTexture("gfx/pete02.png");
13     player->texture = pete[0];
14     SDL_QueryTexture(player->texture, NULL, NULL, &player->w, &player->h);
15 }
16
17 /* the section below implements the new commands for controlling the entity with
18    appropriate control gestures such as WASD or UpDownLeftRight and SPACE
19 */
20 void doPlayer(void)
21 {
22     player->dx = 0;
23     if (app.keyboard[SDL_SCANCODE_A])
24     {
25         player->dx = -PLAYER_MOVE_SPEED;
26         player->texture = pete[1];
27     }
28
29     if (app.keyboard[SDL_SCANCODE_D])
30     {
31         player->dx = PLAYER_MOVE_SPEED;
32         player->texture = pete[0];
33     }
34
35     if (app.keyboard[SDL_SCANCODE_W] && player->isOnGround)
36     {
37         player->riding = NULL;
38         player->dy = -20;
39         playSound(SND_JUMP, CH_PLAYER);
40     }
41
42     if (app.keyboard[SDL_SCANCODE_R])
43     {
44         player->x = player->y = 0;
45         app.keyboard[SDL_SCANCODE_R] = 0;
46     }
47
48     // new code
49
50     if (app.keyboard[SDL_SCANCODE_LEFT])
51     {
52         player->dx = -PLAYER_MOVE_SPEED;
53         player->texture = pete[1];
54     }

```

```

53
54     if (app.keyboard[SDL_SCANCODE_RIGHT])
55     {
56         player->dx = PLAYER_MOVE_SPEED;
57         player->texture = pete[0];
58     }
59
60     if (app.keyboard[SDL_SCANCODE_SPACE] && player->isOnGround)
61     {
62         player->riding = NULL;
63         player->dy = -20;
64         playSound(SND_JUMP, CH_PLAYER);
65     }
66
67     if (app.keyboard[SDL_SCANCODE_R])
68     {
69         player->x = player->y = 0;
70         app.keyboard[SDL_SCANCODE_R] = 0;
71     }
72
73     if (app.keyboard[SDL_SCANCODE_Q]) {
74         messageboxYesNo();
75     }
76 }
77
78 void messageboxYesNo(void) {
79     // referenced from https://wiki.libsdl.org/SDL\_ShowMessageBox#Version
80
81     const SDL_MessageBoxButtonData buttons[] = {
82         { SDL_MESSAGEBOX_BUTTON_ESCAPEKEY_DEFAULT, 1, "Yes" },
83         { SDL_MESSAGEBOX_BUTTON_RETURNKEY_DEFAULT, 2, "No" },
84     };
85
86     const SDL_MessageBoxData messageboxdata = {
87         SDL_MESSAGEBOX_INFORMATION, NULL, "Exit Game", "Do you wish to exit the
            game?", SDL_arraysize(buttons), buttons };
88     int buttonid;
89     if (SDL_ShowMessageBox(&messageboxdata, &buttonid) < 0) {
90         SDL_Log("error displaying message box");
91         return 1;
92     }
93     if (buttonid == 1) {
94         exit(0);
95     }
96     else {
97         return;
98     }
99 }

```

8 Appendix D: Homescreen.c After Modifications

```

1  /* Author: Jason Jay Dookarun
2  Date of Creation: 09.04.2020

```

```

3
4 The following section has been designed as a modification to the existing software
5 designed by Parallel Realities. This addition introduces a home "splash" screen to
6 greet the user to prior to their experiences of the game. Prompts and key shortcuts
7 are introduced to the user to 1) commence the game in a correct manner, and 2) to
8 provide the user with further details on the objective of the game as well as how to
9 play the game via a provided control panel.
10
11 */
12
13 #include "common.h"
14 // The following declarations interlink with common.h and apply the logic and draw
15 // standards followed by
16 // SDL2 libraries.
17 static void logic(void);
18 static void draw(void);
19 // represents message boxes that will be utilised for controls and game exit mechanisms.
20 static void messageboxC(void);
21 static void messageboxYesNo(void);
22 static void messageboxH(void);
23
24 // the following declarations have been utilised and implemented from Parallel Realities.
25 // Link: {https://www.parallelrealities.co.uk/tutorials/shooter/shooter15.php}
26 static SDL_Texture* bckgrndTexture;
27 static SDL_Texture* logo;
28 static SDL_Texture* space;
29 static int reveal = 0;
30 static int timeout;
31 static int existingBackground = 0;
32 static SDL_Texture* background;
33
34 // this section implements the reveal of the contents that are to be displayed on the
35 // screen, and controls that are to be entered by the user
36 static void logic(void)
37 {
38     // this implements the rear wallpaper screen movement if the dimensions of the wallpaper
39     // do not the match the one used.
40     if (--existingBackground < -SCREEN_WIDTH) {
41         existingBackground = 0;
42     }
43     // this counter implements the reveal of the logo that has been utilised
44     if (reveal < SCREEN_HEIGHT) {
45         reveal++;
46     }
47
48     if (app.keyboard[SDL_SCANCODE_SPACE]){
49         // stops existing audio from playing and allows the main stage (i.e. existing
50         // gameplay to load)
51         Mix_HaltChannel(-1);
52         initStage();
53     }
54     if (app.keyboard[SDL_SCANCODE_H]) {
55         // references to messageboxH to show help menu
56         messageboxH();
57     }
58 }

```

```

54     }
55     if (app.keyboard[SDL_SCANCODE_C]) {
56         // references to show controls menu via a message box
57         messageboxC();
58     }
59     if (app.keyboard[SDL_SCANCODE_Q]) {
60         // references to the quit menu via message box prompt(s)
61         messageboxYesNo();
62     }
63
64 }
65
66 /* This draw mechanism has been referenced from map.c and Parallel Realities (link above).
67    This method allows the
68    projection of the main splash screen by drawing the screne via app renders. Once the
69    wallpaper has been rendered, the
70    user is then projected the logo, as well as instructions/controls.
71 */
72 static void draw(void)
73 {
74     SDL_Rect screen;
75     int x;
76
77     // implements the drawing of the wallpaper onto the screen
78     for (x = existingBackground; x < SCREEN_WIDTH; x += SCREEN_WIDTH){
79         screen.x = x;
80         screen.y = 0;
81         screen.w = SCREEN_WIDTH;
82         screen.h = SCREEN_HEIGHT;
83         SDL_RenderCopy(app.renderer, background, NULL, &screen);
84     }
85
86     SDL_Rect r;
87     r.x = 0;
88     r.y = 0;
89
90     SDL_QueryTexture(logo, NULL, NULL, &r.w, &r.h);
91     r.h = MIN(reveal, r.h);
92     loadText(logo, &r, (SCREEN_WIDTH / 2) - (r.w / 2), 250);
93     // projected onto wallpaper as a visible instruction
94     drawText(SCREEN_WIDTH / 2, 450, 255, 255, 255, TEXT_CENTER, "PRESS SPACE TO PLAY!");
95     drawText(SCREEN_WIDTH / 2, 500, 255, 255, 255, TEXT_CENTER, "PRESS C FOR CONTROLS");
96     drawText(SCREEN_WIDTH / 2, 550, 255, 255, 255, TEXT_CENTER, "PRESS H FOR HELP");
97
98 }
99
100 // initiates the loading of the graphics and plays the background audio,
101 void initTitle(void)
102 {
103     app.delegate.logic = logic;
104     app.delegate.draw = draw;
105     // this allows the background wallpaper to be uploaded and displayed for the user.
106     background = loadTexture("gfx/backgroundNew.png");
107     logo = loadTexture("gfx/logo.png");
108     // the music score utilised in the introduction package is an adaptation of Stu
109     Phillips' Knight Rider Theme, covered by

```

```

107     // Jason Jay Dookarun and recorded.
108     loadMusic("music/kr.mp3");
109     playMusic(1);
110 }
111
112 /* MessageBoxH implements a welcome messsagebox, explaining to the user how the game
113    functions in an appropriate manner
114 */
115 void messageboxH(void) {
116     // referenced from https://wiki.libsdl.org/SDL_ShowMessageBox#Version
117
118     const SDL_MessageBoxButtonData buttons[] = {
119     { SDL_MESSAGEBOX_BUTTON_RETURNKEY_DEFAULT, 0, "OK" },
120     };
121
122     const SDL_MessageBoxData messageboxdata = {
123     SDL_MESSAGEBOX_INFORMATION, NULL, "Welcome!", "Hi and welcome To Pete's Pizza Hunt! We
124         have lost 11 slices of pizza and your mission is to find them! To check your
125         progress, look at the HUD on top right-hand side. Good
126         luck!", SDL_arraysize(buttons), buttons };
127
128     int buttonid;
129     if (SDL_ShowMessageBox(&messageboxdata, &buttonid) < 0) {
130         SDL_Log("error displaying message box");
131         return 1;
132     }
133     if (buttonid == 0) {
134         return;
135     }
136 }
137
138 /* messagebox C implements a control method message box to notify the user how to play the
139    game
140 */
141 void messageboxC(void) {
142     // referenced from https://wiki.libsdl.org/SDL_ShowMessageBox#Version
143     const SDL_MessageBoxButtonData buttons[] = {
144     { SDL_MESSAGEBOX_BUTTON_RETURNKEY_DEFAULT, 0, "OK" },
145     };
146
147     const SDL_MessageBoxData messageboxdata = {
148     SDL_MESSAGEBOX_INFORMATION, NULL, "Controls", "Left: LEFT Key or A, Right : RIGHT Key,
149         Jump : SPACE or W, Reset : R, Quit : Q.", SDL_arraysize(buttons), buttons };
150
151     int buttonid;
152     if (SDL_ShowMessageBox(&messageboxdata, &buttonid) < 0) {
153         SDL_Log("error displaying message box");
154         return 1;
155     }
156     if (buttonid == 0) {
157         return;
158     }
159 }
160
161 /* messageboxYesNo implements an exit strategy to exit the game for the client without a
162    forceful termination.
163 */
164 void messageboxYesNo(void) {
165     // referenced from https://wiki.libsdl.org/SDL_ShowMessageBox#Version
166     const SDL_MessageBoxButtonData buttons[] = {

```

```

156     { SDL_MESSAGEBOX_BUTTON_ESCAPEKEY_DEFAULT, 0, "No" },
157     { SDL_MESSAGEBOX_BUTTON_RETURNKEY_DEFAULT, 1, "Yes" },
158 };
159 const SDL_MessageBoxData messageboxdata = {
160     SDL_MESSAGEBOX_INFORMATION, NULL, "Exit Game", "Do you wish to exit the
        game?", SDL_arraysize(buttons), buttons };
161 int buttonid;
162 if (SDL_ShowMessageBox(&messageboxdata, &buttonid) < 0) {
163     SDL_Log("error displaying message box");
164     return 1;
165 }
166 if (buttonid == 1) {
167     exit(0);
168 }
169 else {
170     return;
171 }
172 }

```

9 Appendix E: Outro.c After Modifications

```

1  /* Author: Jason Jay Dookarun
2  Date of Creation: 09.04.2020
3
4  The following section has been designed as a modification to the existing software
5  designed by Parallel Realities. This addition introduces exit screen to. Similar to the
6  homescreen
7  panel, similar methods are implemented with alternative commands added such as a replay
8  method or exit.
9
10 */
11 #include "common.h"
12 // The following declarations interlink with common.h and apply the logic and draw
13 // standards followed by
14 // SDL2 libraries.
15 static void logic(void);
16 static void draw(void);
17 //implements exit message box
18 static void messageboxYesNo(void);
19
20 static SDL_Texture* bckgrndTexture;
21 static SDL_Texture* logo;
22 static SDL_Texture* space;
23 static int reveal = 0;
24 static int timeout;
25 static int existingBackground;
26 static SDL_Texture* background;
27 static int existingBackground = 0;
28
29 // initiates the loading of the graphics and plays the background audio,

```

```

30 void initOutro(void)
31 {
32     Mix_HaltChannel(-1);
33     app.delegate.logic = logic;
34     app.delegate.draw = draw;
35     background = loadTexture("gfx/outro.png");
36     logo = loadTexture("gfx/missionaccomplished.png");
37     // Music from : https://www.bensound.com
38     loadMusic("music/bensound-erf.mp3");
39     playMusic(1);
40 }
41
42 // this section implements the reveal of the contents that are to be displayed on the
43 // screen, and controls that are to be entered by the user
44
45 static void logic(void)
46 {
47     if (--existingBackground < -SCREEN_WIDTH) {
48         existingBackground = 0;
49     }
50
51     if (reveal < SCREEN_HEIGHT) {
52         reveal++;
53     }
54     // restarts the game by transferring the client to the introductory splash screen
55     if (app.keyboard[SDL_SCANCODE_P]) {
56         initTitle();
57     }
58     // similar to homescreen.c, allows the user to meet the exit message box to confirm
59     // their decision.
60     if (app.keyboard[SDL_SCANCODE_Q]) {
61         messageboxYesNo();
62     }
63 }
64
65 /* This draw mechanism has been referenced from homescreen.c, map.c and Parallel Realities
66    (link above). This method allows the
67    projection of the main splash screen by drawing the scene via app renders. Once the
68    wallpaper has been rendered, the
69    user is then projected the logo, as well as instructions/controls.
70 */
71
72 static void draw(void)
73 {
74     SDL_Rect screen;
75     int x;
76
77     for (x = existingBackground; x < SCREEN_WIDTH; x += SCREEN_WIDTH) {
78         screen.x = x;
79         screen.y = 0;
80         screen.w = SCREEN_WIDTH;
81         screen.h = SCREEN_HEIGHT;
82         SDL_RenderCopy(app.renderer, background, NULL, &screen);
83     }
84     SDL_Rect r;
85     r.x = 0;
86     r.y = 0;

```



```

82
83     SDL_QueryTexture(logo, NULL, NULL, &r.w, &r.h);
84     r.h = MIN(reveal, r.h);
85     loadText(logo, &r, (SCREEN_WIDTH / 2) - (r.w / 2), 250);
86     // projected onto wallpaper as a visible instruction
87
88     drawText(SCREEN_WIDTH / 2, 450, 255, 255, 255, TEXT_CENTER, "PRESS P TO PLAY AGAIN!");
89     drawText(SCREEN_WIDTH / 2, 500, 255, 255, 255, TEXT_CENTER, "PRESS Q FOR QUIT");
90
91 }
92
93
94 /* messageboxYesNo implements an exit strategy to exit the game for the client without a
95    forceful termination.
96 */
97 void messageboxYesNo(void) {
98     // referenced from https://wiki.libsdl.org/SDL\_ShowMessageBox#Version
99
100     const SDL_MessageBoxButtonData buttons[] = {
101         { SDL_MESSAGEBOX_BUTTON_ESCAPEKEY_DEFAULT, 0, "No" },
102         { SDL_MESSAGEBOX_BUTTON_RETURNKEY_DEFAULT, 1, "Yes" },
103     };
104
105     const SDL_MessageBoxData messageboxdata = {
106         SDL_MESSAGEBOX_INFORMATION, NULL, "Exit Game", "Do you wish to exit the
107             game?", SDL_arraysize(buttons), buttons };
108
109     int buttonid;
110     if (SDL_ShowMessageBox(&messageboxdata, &buttonid) < 0) {
111         SDL_Log("error displaying message box");
112         return 1;
113     }
114     if (buttonid == 1) {
115         exit(0);
116     }
117     else {
118         return;
119     }
120 }

```
