

ECE 2564: Project 2

PROJECT COMPLETED BY LLA ROCHEZ

June 28th, 2021

VIRGINIA TECH | GITHUB ID: ILLAR47

Report Summary





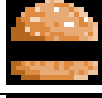

The purpose of this report is to summarize the functions and capabilities of “Dinner Dash”, which is the title of this Project 2 submission. In addition, the report will outline the project description, joystick implementation, code quality, and the bonus features of “Dinner Dash”.

Project Description

All of the tasks outlined by the initial project description can be accomplished by this game. This game has the player take on the role of a chef who runs a restaurant. When the game is first booted, a title screen with the game’s name and the creator’s name is displayed for three seconds. After three seconds the title screen will disappear, and a menu will appear on the screen.

When the menu appears, the user can navigate the menu by “tapping the joystick up or down. The cursor that initially points to “How to Play” will move up or down depending on the direction of the joystick movement. The cursor moves only once per “tap”. The movement of the cursor is circular. If the user pressed the Joystick button, the game will navigate to the option selected. Selecting How to Play opens the instructions. Pressing the Joystick button again returns the user to the menu. Selecting High Scores opens the scores page. This page displays the top three scores. These start at \$0 and as new high scores are achieved, the scores will update. New high scores are achieved by playing the game. Like the instructions screen, pressing the Joystick button will return the user to the menu.

If the user has selected “Play Game” the gameplay area will open. This area has an initial score of “\$0”, a timer of 30 seconds, and a display with selectable ingredients. The ingredients are listed below:

Ingredient (Names were modified to better fit on display... or because it added humor)	Picture
Pattty (Patty)	
Spinch (Lettuce)	
Cheese (Cheese)	
Tomat (Tomato Slice)	
Bunnn (Bun)	
Soauce (Sauce / Condiments)	

Additionally, there are the following features:

- The **Income** received (top left of the screen)
- The **Time** remaining (top middle of the screen)
- **Ruined Orders** (invisible when the game first starts, top right of screen. These will be indicated by Xs)
- The current **Order / Ticket** (with ingredients requested, bottom of screen)

The order ticket has a list of requested ingredients along with an order name. The table below lists the possible recipes:

Order Name	Ingredients
Boring Bob	Patty
The Vegetarian	Cheese, Lettuce, Bun, tomato
Fully Involved	Patty, Cheese, Lettuce, Bun, Condiments, tomato
The salad	Cheese, Lettuce, Tomato
Manly Cheesy Burg	Cheese, Patty, Bun, Condiments
Basic Boi	Bun, Patty
The Weirdo	Cheese, Bun
What the Hell?	Lettuce

The order that the ingredients are selected does not matter. The user can push the joystick in the direction of the ingredient they wish to select, and a green box will highlight the selection. The user can press S1 button to select the ingredient, or S2 button to deselect the ingredient. The user can press the Joystick button to submit the order. If the submitted order is correct, the player will increase the score / income (in the upper left corner) by \$5. If the submitted order is incorrect, the player will receive a strike in the upper right corner of the screen. Gameplay ends when the player has three strikes, or the timer has run out. A game over screen will be displayed. If the player achieves a new high score, the screen will be different. Pressing the Joystick button will return the player to the menu.

Additional Art used (Made by creator):



State Visuals:



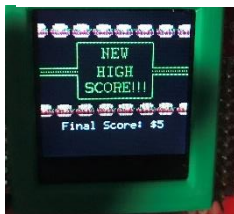
G Gameplay selections



F Gameplay start state



E Gameplay additional start state



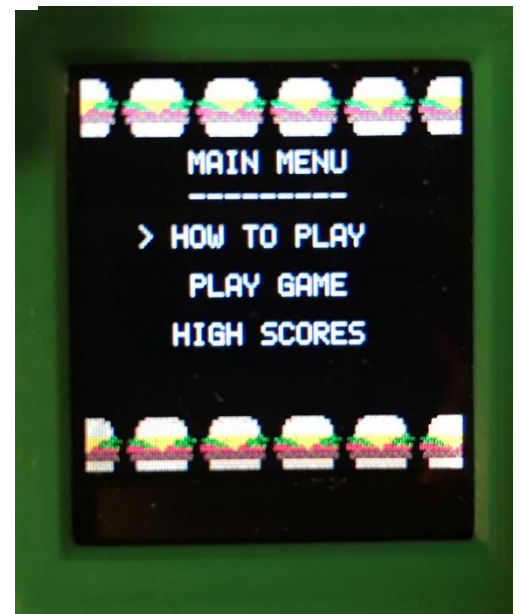
D Success Game over screen



C Title Screen



B Gameplay highlighted option



A Main Menu Screen

Using the Joystick

The Joystick was used to implement controls for a few sections. General methods were developed to determine if the Joystick was tapped in any specific direction. These methods were then used to implement the radial cursor as well as the highlight, deselection and selection methods used for gameplay. The X position of the joystick was also used to generate the recipes randomly.

For the radial cursor, the method checked if the joystick was pushed up or down and responded accordingly. A similar method was used for the highlight method. The direction check was used in conjunction with a check for if either S1 or S2 were pressed to implement the selection and deselection methods. The randomization was implemented by using the rand method. Srand() was called first, with

the x position of the joystick as its parameter. Rand() was then called and modded (%) by the number of recipes. This ensured that the random numbers would only vary from zero to the number of recipes.

Code Quality

The aspects that had to be adhered to are as follows: firstly, the code needed to be reasonably commented for understandability and ease of grading for the TA. Additionally, #defines were to be used. Hardcoded numbers were not permitted on this assignment with the exception of GFX calls. Functions were not to exceed the length of 50 lines of code, within reason. This number does not include comments, brackets, and other non-code related statements. Global variables were also not permitted, and the code needed to be non-blocking. The table below outlines these requirements, the expected score and the reason for the expected score:

Code Quality Requirement	Expected Score	Reason for Expected Score
Code is reasonably commented so that a TA can understand your code.	10/10	Comment at the head of each method with parameters included. In addition smaller comments within the body to clarify and explain what is occurring and when it is occurring.
#defines and constants are used instead of hard-coded numbers.	20/20	Defines are used in Application.h and no hardcoded numbers are used, with the exception of Graphic statements.
No function exceeds 50 lines of code, within reason	30/30	All functions fall below this maximum.
No global variables are used, with the exception of possibly custom images using the Image Reformer.	20/20	No global variables were implemented, excluding custom icons, backgrounds, ect.
Your code is completely non-blocking. Pressing LB1 lights up Launchpad LL1 at any time.	20/20	At all points, LB1 is able to be lit.

Bonus

For the bonus features, an additional 5 recipes were implemented along with an additional 3 ingredients. Artistic features such as a custom display and icons were also implemented. To generate random numbers, the x coordinate of the joystick was used to randomize the outputs of rand(). It is uncertain if this is what was meant by the bonus prompt, however it will be included to ensure a detailed report. See the table of features along with the expected points below:

Bonus Feature	Points expected
Custom Images	25/25
More Recipes (Max: 5 additional recipes)	25/25
More Ingredients (Max 5 Added)	30/50
Using Joystick to generate random numbers	? / 50