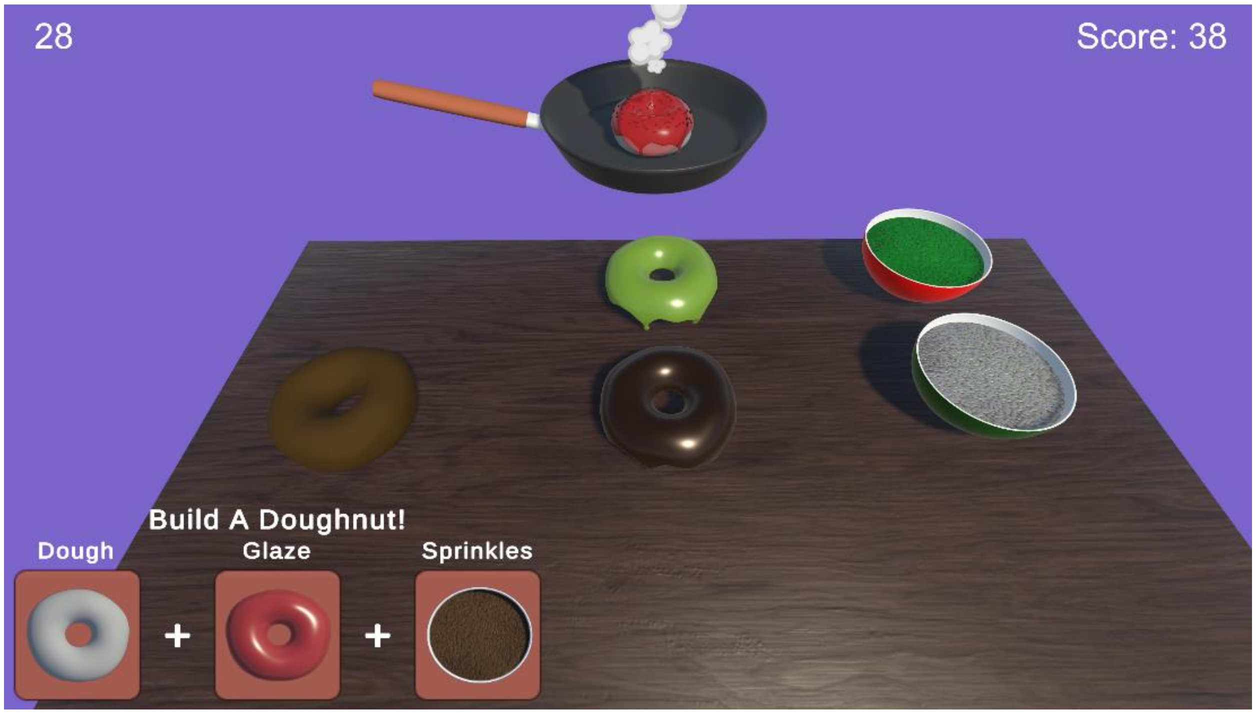
Doughnut Chef Documentation

Evan Svendsen



Welcome to doughnut chef; you are the best Doughnut Chef. You will race against the clock to test your culinary doughnut skills. Complete recipes and gain points before the clock runs out.

You will have to complete the appropriate recipe by selecting 1 dough, 1 glaze topping, and 1 color of sprinkles. Each doughnut consists of these 3 parts. Ingredients can be added in any order to the frying pan. There will be 2 dough options available for each game selected randomly from a list of 4 available dough bases: Plain, Chocolate, Strawberry, and White Powder. There will be 3 glaze options selected from a list of 6 total: Chocolate, Maple, Matcha, Plain White, Strawberry, and Chocolate. And there will be 3 options for sprinkle colors selected also from a list of 6 colors randomly: Blue, Brown, Green, Pink, White and Yellow.

Recipes will be generated automatically when certain game events happen. At the start, at the completion of cooking, when the pan is clicked and it is not cooking, and when the incorrect ingredients are selected. When the 60 second timer runs out the game will be over, and your total score will be displayed.

Clicking on the pan while the doughnut is cooking, and the correct ingredients have been selected will result in a 20% bonus on the recipe score earned. Clicking the pan before the ingredients have been added will result in a -5 penalty to the score. No additional ingredients can be added to the pan while the doughnut is cooking. The pan will automatically remove all ingredients in the event of the incorrect ones being selected. You must select 3 ingredients for the recipe to be compared.

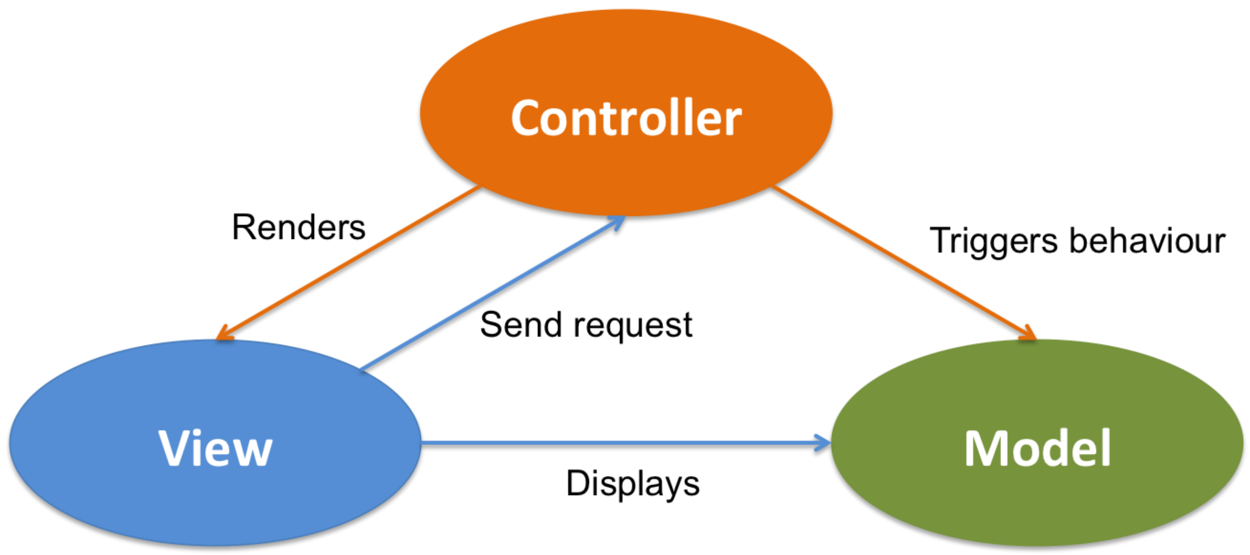
Game loop is complete, and you can restart the game when it is over from the game over screen.

Gameplay Flow

- The game will start, and a clock timer will count down from 60 seconds running asynchronously in the background.
- The game will select all available ingredients to be rendered onto the table. 2 doughs selected from an available 4. 3 glaze options selected from 6, and 3 sprinkle options from a total of 6.
- Once the available ingredients are generated, they will be rendered on the table, and a new recipe will be generated.
- The HUD will update each time a new recipe is generated to show the required base, glaze, and sprinkle combination requested by the customer.
- The player can click on ingredients in any order. Clicking on an ingredient will cause it to disappear and show on the surface of the frying pan. Hovering over any ingredient will show visually that it has been hovered over, as well as playing a sound.
- Once 3 ingredients are in the pan, the frying pan will compare with the **GameManager** that holds the current recipe. If the recipe is not a match, then an error sound will play, and -5 points are subtracted from the player.
- If the recipe is a match, then the cooking timer will automatically be started. Cook time for each recipe is randomly generated between 3 and 8 seconds. A circular pie-chart cook timer will be shown on screen to indicate to the player the amount of time the cooking will take. At exactly halfway done the cooking process, the 3 ingredients in the pan will be hidden, and a singular doughnut that is the final product will display instead for the remaining 50% of cooking time.
- Hovering over the frying pan will play a sound and indicate to the player that they are selecting the frying pan.
- If the frying pan is clicked while the cooking timer is active, then an additional 20% bonus to the score will be added from this recipe. A special bonus sound is played.
- If the frying pan is clicked before all ingredients have been added, then a -5 point deduction will be given, and a new recipe is generated. All ingredients are cleared from the pan, and the table is reset. An error sound is played to indicate to the player an incorrect ingredient selection.
- While the recipe starts cooking, the sound of frying will play in the background and stop when done. The sound will also stop if the game ends before the cook time.
- No points are awarded for recipes that do not cook in time before the clock runs out.
- Once the recipe is complete cooking, then the score will be added to the total score for this session. Score for each recipe is randomly generated in multiples of 5 between 10 and 30.

- Once the recipe is cooked, then the pan will be emptied, the table is reset, and a new recipe is automatically created. A sound will be played indicating to the player that cooking is complete and a new recipe is generated.
- When the timer runs out and reaches 0, the Game Over screen will be shown and the player can restart a new game with randomly selected available ingredients. And end of game sound is played when the game over screen is shown.
- Music will play on continuous loop in the background.

Architecture



The code for the project is loosely based on the Model, View, Controller scheme (MVC) but does not 100% adhere to its structure. Some views and visuals of objects are managed by the controller class for that object as well. Such as the frying pan controlling the view of the ingredients in the pan. An additional namespace for Helpers is included for classes that do not easily fit the MVC scheme. The code for the game can be found under **VXR1170 > Code Prototype > Scripts**. With additional scripts added to **Shared > Scripts** that can be used across multiple projects and assignments.

Game events are broadcast from a static class called **GameEventBroadcaster**. These are global events that any script can subscribe to. This is done to avoid coupling dependencies together, and it makes the code more modular for testing in isolation.

Game Logic

- Recipe Selection: Pick 3 ingredients at random 1 from each of the following ingredient categories: Dough, Glaze, and Sprinkles.
- Player Input: Use trigger colliders to detect player mouse over objects.

- Score: Only incremented when the recipe is correct and when cooking has completed. Value of earnings will be a randomly generated multiple of 5 between 10 and 30.
- Use empty transforms to get the positions on the frying pan where to display the selected ingredients.
- If the frying pan is full, then prevent clicking additional ingredients.
- If the frying pan is not full, prevent submission of the recipe and reset the recipe from occurring.
- If the frying pan is clicked while cooking, add bonus score
- If the frying pan is clicked before cooking reduce the player score by 5.
- Remove the ingredients from the frying pan after each new recipe is generated.
- If the timer reaches 0, the trigger game over events.
- The timer will always reset to 60 for each new game.
- If the game is reset from the game over screen, then trigger new game events.

<u>Code</u>

Shared Namespace:

This namespace contains scripts that can be used across multiple assignment projects. (if a script in the Shared folder is not listed, then it is not being used in this project).

Singleton – An abstract class that allows global access to singleton scripts. That is scripts with only a single copy in the game scene.

Timer

WaitForFrame() -> asynchronous task that waits for 1 frame before continuing.

WaitForSeconds(float) -> asynchronous task that waits for the given number of seconds before continuing.

WaitUntil(Func<bool>) -> asynchronous task that waits until the given condition becomes true before proceeding.

ExtendedEnumerables

A C# extension class that adds support for lists arrays and other enumerables.

SelectRandom(this IEnumerable<T>) -> Randomly selects 1 item from a list of available items.

Core Namespace:

This namespace contains core elements of the overall code.

Camera Exception Class - Custom exception for the **CameraMover** script.

Controllers Namespace:

This namespace manages and controls all game behaviour and flow.

Frying Pan

Highlight(bool) -> Will highlight / de-highlight the frying pan by toggling the sphere around it. Will play a sound if highlighted.

AddIngredient(IngredientType, int) -> Called by the event broadcaster. Adds the ingredient into the frying pan if not full and GameOn is true from the game manager. Pan class handles the visual display of the newly added ingredient. It will check if the pan has all ingredients and submit it to the **GameManager** class for review.

ClearPan() -> Remove the ingredients displayed from the frying pan. Sets isCooking to false and resets the bonus modifier. It will deactivate the smoke and stop the cooking routine.

SubmitRecipe() -> If the pan is full of ingredients, call Game Manager's **CheckRecipe** method. If the recipe is a match, begin cooking otherwise negate 5 points from the score and call **ResetRecipe** method of the GameManager.

CookDoughnut() -> Called when the recipe is correct. Active the smoke particles and start the countdown timer for the cooking. Wait for half the time to cook before hiding ingredients in pan and then showing the final product. Once cooking is complete, only update the score if the game is still running. Calculate the player score based on a 20% bonus if the pan was clicked during the cooking process. Call **ResetRecipe** of the GameManager**.**

CompleteDoughnut() -> Handles the additional steps of showing the final product in the pan.

Game Event Broadcaster

Handles the broadcasting of 3 game events

- When the game is started and the available recipe ingredients are chosen.
- When an ingredient is clicked and added to the pan.
- When a new recipe is generated.

Game Manager

StartNewGame() -> Resets the score to 0, Sets the game on to true, and starts the countdown timer after generating a new recipe.

Countdown() -> Remove 1 from the countdown time ever second if the game is active and update the UI calling the HUD.

ResetRecipe() -> Uses the recipe **GenerateRecipe** method to select a new recipe. Calls **ClearPan** method of the frying pan. Call **ShowRecipe** on the HUD class to visually display the new recipe. Broadcast that a new recipe is generated using the **GameEventBroadcaster**.

CheckRecipe(List<Tuple<IngredientType, int>>) -> If the ingredients in the pan match the ingredients of the recipe, return true to the frying pan otherwise false.

Update() -> Check if the timer has reached 0, call **GameOver**

UpdateScore(int) -> Update the score and call the HUD to update the visuals.

GameOver() -> Block player input. Stop the countdown timer coroutine. Show game over screen.

Ingredient

int ID -> Unique identifier of the ingredient.

IngredientType category -> the category this ingredient belongs to.

HighlightIngredient(bool) -> Highlights/de-highlights the ingredient and plays a sound.

OnMouseClick() -> When mouse is clicked on the ingredient call the **AddIngredient** method of the frying pan using this ingredient's ID.

Data Namespace:

This namespace manages all data models for the game, and it controls no game behavior.

Constants

Global static class of constants that can be configured from 1 location.

IngredientType

Enum for each category of ingredient.

Recipe

Data container for recipe information.

RecipeBuilder

Generates a random recipe from the list of available ingredients on the table.

Views Namespace:

This namespace manages all scripts that control visual aspects of the game.

Camera Mover

Used to move the camera over the correct ingredient items being rendered to the render texture in the UI.

Cook Timer

Used to run the pie chart timer for the cooking time.

HUD

UpdateScore(int) -> Update the value of the score on the game canvas.

UpdateTimer(int) -> Update the value of the time on the canvas.

ShowRecipe(List<int>) -> Given the list of ingredients, show the recipe in a visual way to the player.

RandomMaterialLoader

Used to load a random material for the sprinkles bowls to make the game more dynamic

Helpers Namespace:

This namespace manages all scripts not easily fit into the above categories.

Sound Manager

PlayClip(AudioClip) **->** Plays the given audio clip on the source**.**

StopSound() -> Stops all sound on the source.