Cake Blob - Phase 4 Report

Instructions for Jar file and Javadocs:
Jar File location: Blob2DGame/Blob2DGame.jar
JavaDocs Html: Blob2DGame/target/site/apidocs/

The Game:

Our game is titled “Cake Blob”, however the original name we gave it was Blob 2D Game. The lore of our game surrounds a young blob’s cake addiction and the depletion of his parents' cake factory. Our protagonist Jimmy, runs around the factory eating slices of cakes, whilst running from his parents to ensure he can increase his sugar levels. However, his parents are extremely concerned about Jimmy’s health, as a result, to aid in preventing little Jimmy from eating too much sugar, they implemented a freeze and teleportation trap to keep Jimmy away from the slices of cakes. If Jimmy is able to successfully eat all the cakes before the time runs out and reach the exit before his parents catch him, Jimmy wins!

In terms of our devotion to the original lore, we believe we kept true to the core aspects of the storyline. We implemented all of the features we declared during the first planning phase, and attempted at keeping consistency with vocabulary, such as rather than using score or health, we use Sugar Levels to indicate our protagonists score and health in one. However, we decided to exclude the flash light feature, as it would increase the difficulty significantly, as well as the mechanics for the enemy become overly unpredictable. Thus, we exchanged the flashlight feature for another teleportation trap, which sends the protagonist to the beginning if stepped on.

In contrast to our devotion to the lore, we were unable to keep loyalty to the original UML diagram we had planned ahead. At the beginning of this project, our group was inexperienced in Java and OOP programming, this led us to having a weak UML diagram at the beginning, lacking some core features and aspects to a proper OOP structure. As a result, we have reformed our coding structure so that our methods and classes are reusable and flexible. Although we tried to keep along the lines of our original UML diagram, we had to add more classes that are referenced to the main class, such as the GameBoard class, which references our main class Game (this class handles the JFrame and JWindow). GameBoard was implemented to handle all of our game features objects; furthermore, we organized all of our in game entities under the entities class, which we had planned to divide into two parts, Non-moveable and move-able entities to further organize the coding structure, but was not necessary.

Furthermore, our interface was altered slightly as well, we kept one of our original interface templates but modified them for the final version.

In relation to our original project plan (for meetings/communication), we followed our initial structure relatively well. We met about 2-3 times per week, however this varied depending on what component was due and how close we were to the due date. This number was closer to 1 if we didn't have any deadlines coming up, and perhaps even 3 or 4 if we were approaching a due date. To communicate we stuck to using Discord and WhatsApp as discussed in our original plan - then for collaborative work we just used our own branches on the shared repository for code, then Google Docs for any written deliverables. In terms of work division, usually we would not just take large tasks on separately, but instead it seemed as if we would all just work on small chunks depending on what needed to get done first (selected arbitrarily), then update the group on what we did once finished, and check what we needed next. This process worked for us because we tended to get large chunks of our work done during joint meetings where we were all present. In times where we couldn't meet consistently to work all at the same time we tried to merge changes regularly as well as still communicate to the other group members what we were changing/working on, so that there were no surprises once they started working on it.

TakeAways:

Throughout this project, our group has learned that having a plan helps speed up the production process significantly, as we have a starting base and a roadmap to guide us towards our end product. This was our first time using JUnit to test our code, this process taught us how to find bugs, for example one we found during our testing phases was that when we ran our game, there would be multiple instances of our entities, and at every tick of the game, the number of entities would multiply by 2. Therefore whilst we see one player and 2 enemies, there are actually several enemies and players layered on top of eachother, which led to extreme performance deficiencies when our team playtested.

This project has aided our group in gaining proficiency in Object Oriented Programming, as we were able to practice implementing both inclusion and overriding Polymorphism when structuring our code. This allowed us to reduce redundancies and create a more efficient layout where we could easily implement new features to our game without restructuring our code as whole. In addition, the testing phase allowed our group to practice aggregation, as we needed to create our class methods to be more

flexible and reusable, so by parameterizing them, we were able to reuse the methods rather than having to recreate/duplicate and create redundancies.

We've also learned how important it is to maintain constant communication between us, as well as push changes that we make regularly. When we were just starting to implement our game design, back in phase 2, it seemed that we had the habit of trying to complete whole functions or features in "one-go", then pushing that to our master branch. We essentially learned the hard way that this was not the best practice as we had to deal with many confusing merge conflicts and other logic issues early on before getting better at consistently updating the master branch with our changes. The communication component comes in when understanding the thought process of other group members when they were implementing something, as many features were built off of, or were related to one another.

Tutorial:

Understanding the game characters and objects

When you load up our game, the first thing that you will see is the title screen (shown below):

As you can see, all of our game's characters, objects, etc. are shown here, along with a brief summary of what they do. You will be playing as the green “Blob - Player” shown first on the left column, and can simply use your “WASD” keys to move up/left/down/right. You can also pause the game at any time using the “P” key.

In addition to your player, there are two enemy characters, your mom and dad blob, which are the pink and red “Blobs” - who will be chasing you. Don’t let them catch you or you will lose the game immediately!

As you play, you must also watch out for two types of traps that have been set up. The freeze trap (shown as a snowflake) will both stop you from moving for one second, as well as decrease your sugar level by 100 (you can think of this as your score). And the

teleport trap (shown as a teleport vortex circle - the last image in the left column), will both send you back to where you started the game (position wise), as well as reduce your sugar levels by 50.

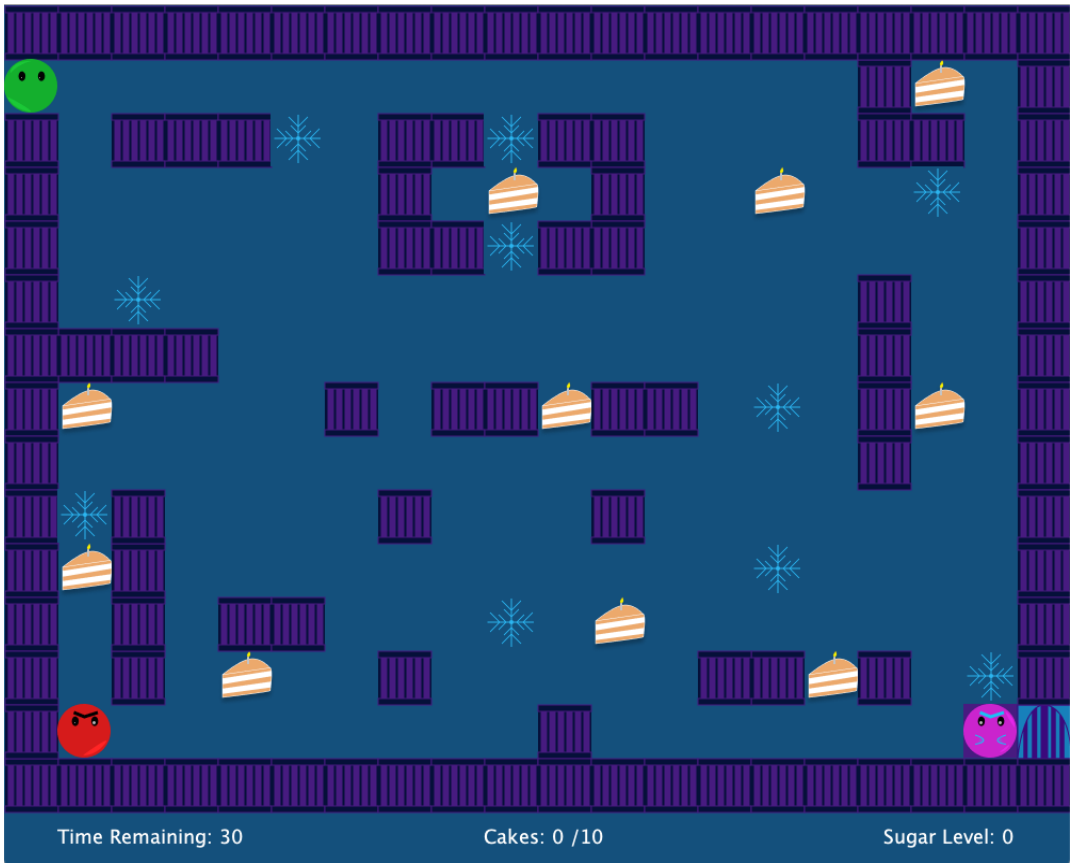
As you are running around you must collect all ten cake slices to unlock the gate. The cake slices will also increase your sugar levels by 100 (for each slice), and there is one bonus cake that will appear randomly throughout the game, which will multiply your current sugar levels by 1.5 and increase it by 200 on top of that (These are both shown in the right column).

The gate will be closed normally, but will open once you collect all ten cake slices, as stated previously - the images for these states are also shown in the right column (below the cakes). You can also see what the frozen animation will look like if you run into a freeze trap (last image on right column).

Remember, the objective of the game is to collect all of the cake slices and reach the exit gate before your mom and dad “Blob” can catch you, and before the timer runs out (will count down from 30 upon starting the game). You will also lose the game if your sugar levels get below 0 - recall that this can be boosted by collecting cakes, but decreased by colliding with the traps.

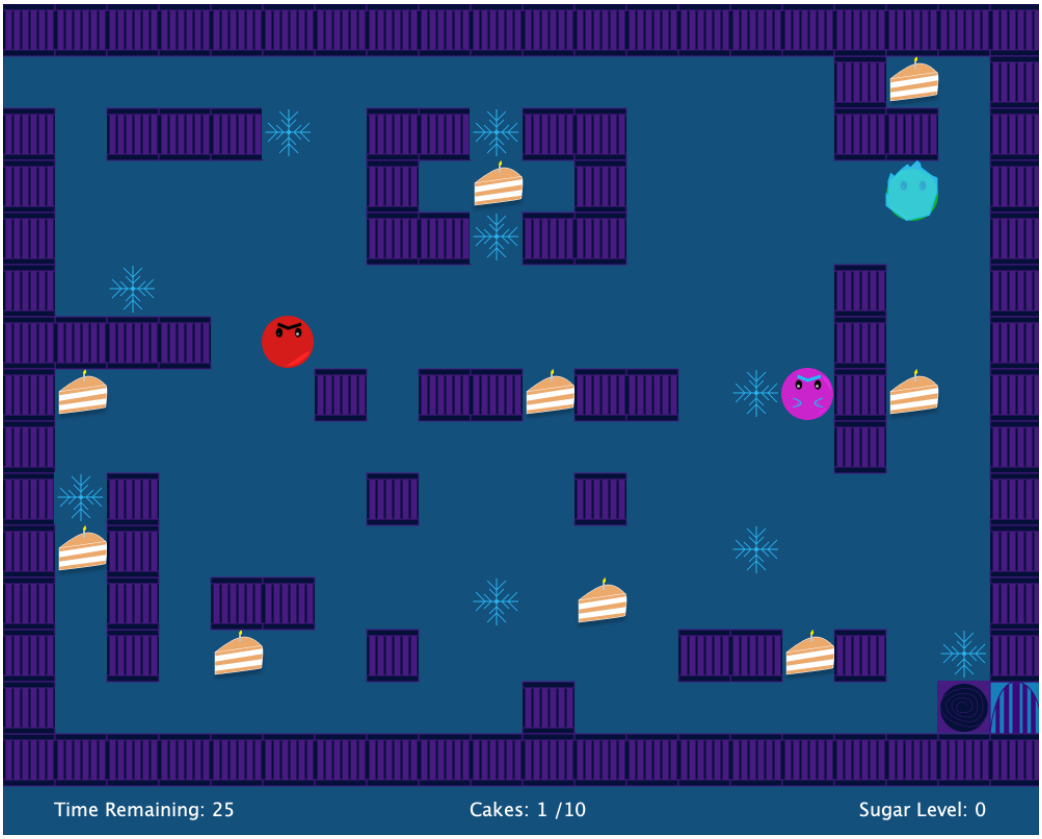
Starting the game

When you are ready to start the game, click the black “START” button to jump in; upon doing so you will see the following display (shown below):



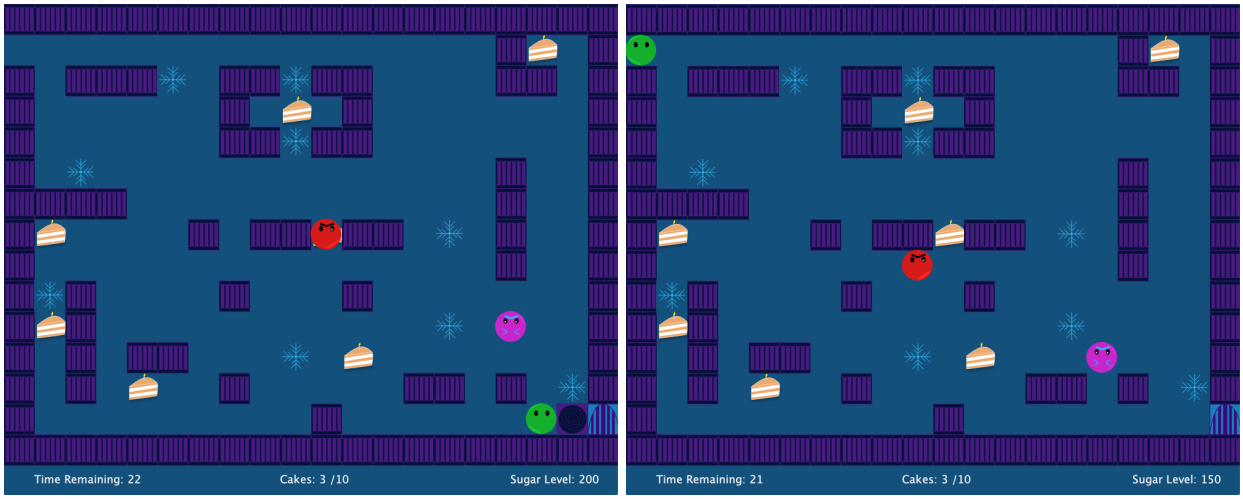
Now you can see the layout of the map. The various enemies, traps, and cakes are placed throughout - also the purple blocks are the walls of the map, i.e. you can’t move there, and neither can the enemies. Notice at the bottom of the screen your time remaining, cake count (/10), and sugar level are displayed.

Freeze mechanic



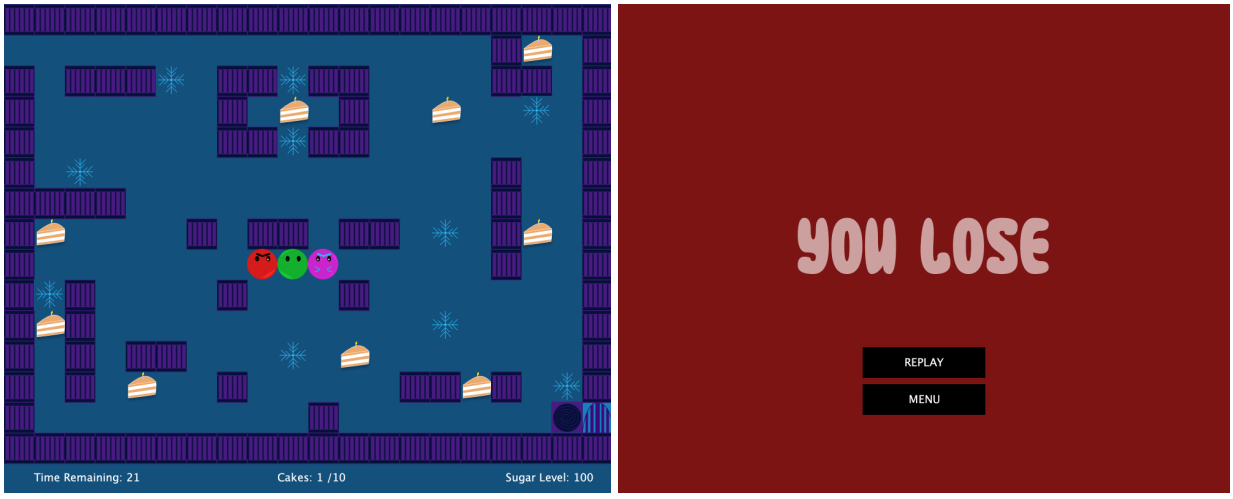
If you happen to run into a freeze trap (and you have a sugar level greater or equal to 100 so you don't lose immediately), your player will be unable to move for one second, and as you can see, you will be stuck in a block of ice. During this time, the enemies will continue to move towards you and the game will still end if they catch you during this freeze state, so you must be careful not to get frozen when the enemies are close behind. Also, remember that the freeze trap decreases your sugar level as well.

Teleport mechanic



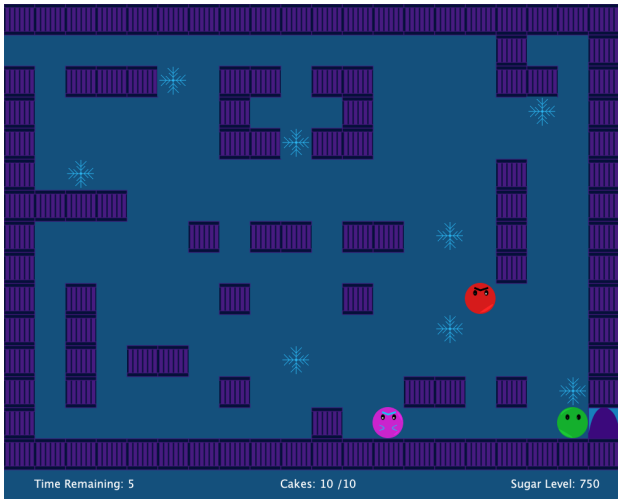
As you can see in the two screenshots shown above (which show the before and after running into a teleport trap), if you happen to run into this trap you will be sent back to the tile in which you started the game. Your sugar level will also decrease, so beware of having too low of a sugar level before running into this trap.

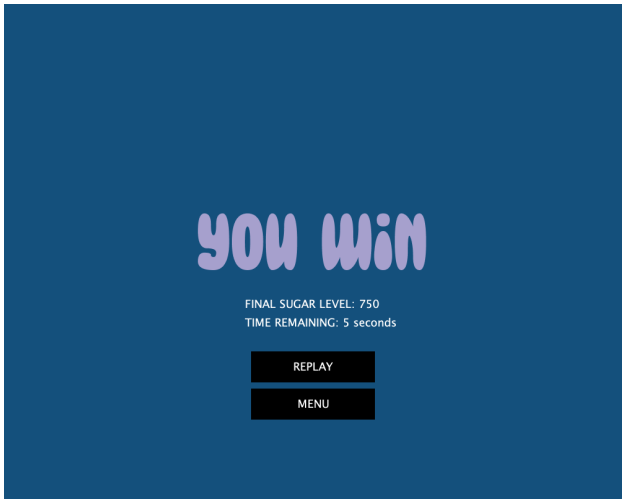
Losing the game



If you get caught by either enemy (which is the situation shown above), or if your sugar level gets below zero (by running into a trap); you will lose the game and be shown this lose screen. Once here you will have the options to both replay (start the game over), or return to the menu screen (where you can see the game characters, objects, etc. again).

Winning the game





If you are able to collect all of the cakes without allowing your sugar level to get below zero, or either of the enemies to catch you, the gate will be opened - and you will win the game if you can reach it before the time runs out. On the win screen your final sugar level will be displayed, as well as the time remaining upon completion. As in the lose screen, you have both options of replaying the game and returning to the main menu.

Pause feature

Remember that at any point during the game, if you want to pause you can press the “P” key and all the enemies, as well as your player will be stopped from moving. The time remaining will also be paused, and the word “PAUSED” will be displayed in the centre of the screen to let you know what state the game is in.