Final Project - Report
Project: Dodge

CS174A - Professor Asish Law

Github repo: https://github.com/dylanphe/dodge.git

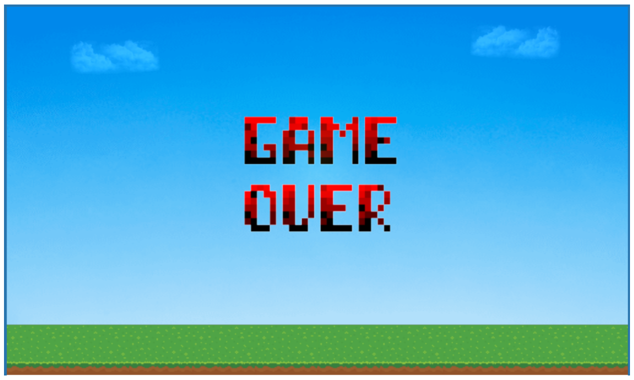
I. Team Members:

❖ Dylan Phe **UID**: 505-834-475 **Email**: dylanphe@g.ucla.edu
❖ Ming Chen **UID**: 705-845-642 **Email**: ming95@g.ucla.edu
❖ Brandon Tran, **UID**: 705-830-462 **Email**: tranbrandon1233@gmail.com

II. Background and Development:

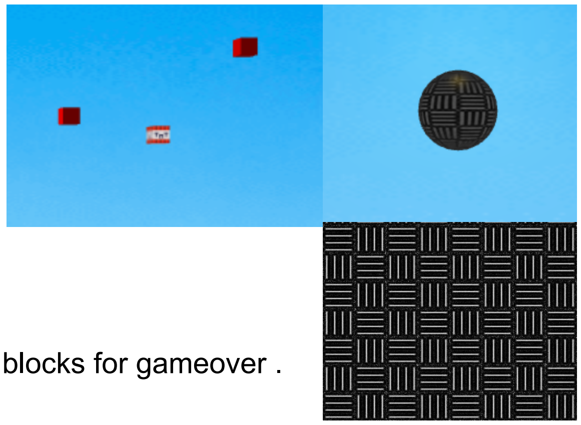
Inspired by the video game of the same name by Atridge, the proposed project, Dodge, is a computer graphics game created with TinyGraphics based on the famous video game of the same name by Atridge created for the Pico-8 game engine in 2019. Although both games follow the same instruction in which a player is to stay alive as long as possible by dodging the blocks and accumulating more scores, our version of the game is a little different in terms of designs, controls and levels of difficulties.

III. Design:



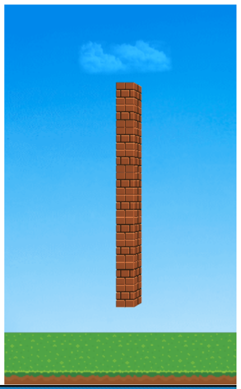
For the game environment, we chose a classic look with a super-mario like design to give our game a vintage look. To light up the game environment, a point-light source is placed near the origin of the world space in the color that matches the Sun. In addition, many textures were also used to distinguish between different types of obstacles given that most of them are of the same dimension. The game is implemented to have a 2D visual by placing the camera position at one location and fixing their screen parameters in both the x and y directions defining the edges of the screen.

IV. Game Components: Characters

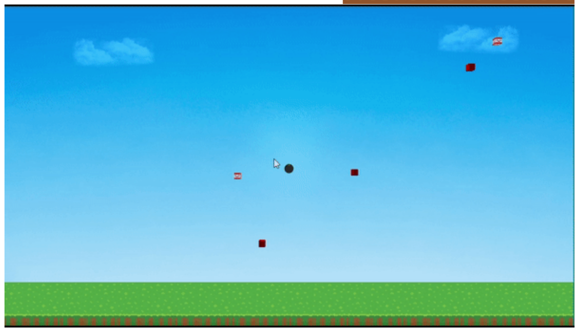


- **Player**: A black dodge ball.
  - **Material**: Phong_texture shader
  - **Keyboard-Interactivities**:
    - **Arrow-Up**: Move up
    - **Arrow-Down**: Move Down
    - **Arrow-Left**: Move Left
    - **Arrow-Right**: Move Right
  - **Features**: Collision detection with other blocks for gameover .

- **Small Blocks**:
    - **Material**: Phong shader
    - **Logistics**:
        - **Levitate around** within the screen parameters
        - **Bounce back** when it hits the edges of the screen.
        - **Change its direction toward the player** if it is detected to be within a certain radius from the player, which is calculated using the Euclidean distance formula. Similarly, the change in direction was also made to be quick enough that it won't keep following the player during the period it was within the targeted radius distance from the player that would lead to an eventual collision.
    - **Features**: Collision detection for score update, game over, explosion effect, and the bouncing off effect.
- **Gliding Brick Blocks:**
    - **Material**: Phong_texture shader
    - **Logistics**:
        - **Appear randomly at different intervals** based on the level of difficulty.
        - **Glide in** from either the left edge screen or the right edge of the screen.
        - **Collision detection** with other blocks for the bouncing off effect, explosion effect and game over.



- **Explosive Blocks**:
    - **Material**: Phong_texture
    - **Logistics**:
        - Explode when colliding with other small blocks, brick blocks or other explosive blocks.
        - Explosions can hit the player.

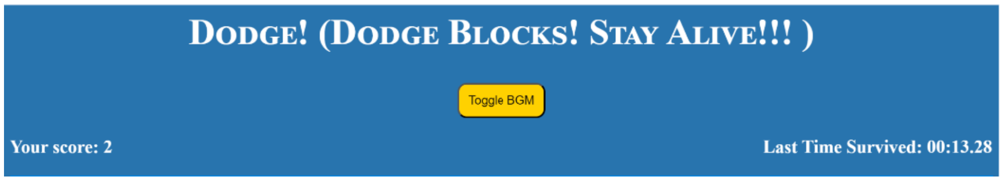


Note: we made the player invincible to capture this gif

V. Gameplay: Score and Difficulty Scaling

- **How do players earn a score:** Players are expected to dodge the blocks in a way that they are also guiding them to collide with one another but not with the player themselves.
    - **Two points** will be earned if:
        - Two small blocks collide.
        - A small block collides with a small explosive block.
    - **Three points** will be earned if small explosive blocks collide with one another. (Bigger explosion)

- **Game difficulty scaling**: The game difficulty depends on the amount of scores the player has accumulated during the gameplay.
    - **Score >= 0**: 5 small blocks at a slow speed.
    - **Score >= 6**: Brick blocks start appearing from the left and glide to the right.
    - **Score >= 10**: Brick blocks start appearing from the right and glide to the left.
    - **Score >= 16**: Small blocks increase speed.
    - **core >= 25**: Three small explosive blocks are added and two small blocks are taken out.
    - **Score >= 30**: Brick blocks generated at a faster rate.
    - **Score >= 40**: Two small blocks are put back in.
- **Game result**: The score will be updated live as shown on the webpage below. At the end of every game, we also showed the amount of time that the last player survived. A button is also provided to play or pause the background music.



DODGE! (DODGE BLOCKS! STAY ALIVE!!!)

Your score: 2

Last Time Survived: 00:13.28

VI. Code and Demo:

- Clone git repo: **https://github.com/dylanphe/dodge.git**
- Located in the project folder, you can click on host.bat for the window platform or host.command on the MACOS platform to start the server at port 8080. Alternatively, you can also use an IDE such as WebStorm to host the game.

During the demo presentation, we want to clarify that we modified the conditions of the game for the sake of presenting. The game score was initially set to 25 so that the audience can see all obstacles during the demo. Similarly, we also implemented an extra button to move the player forward so that it doesn't collide with other blocks when presenting. The actual game doesn't start like that nor have a button to move the player forward in z direction, given that we are implementing a 2D game with 3D visuals.

VII. Challenges faced:

- The most challenging part of the whole project is to determine the path for these obstacles since each one of them has different behaviors and spawned at different times. We have to ensure that their movement looks natural, smooth, and most importantly RANDOM.
- Second, these obstacles also need to be generated following a random pattern. We weren't able to achieve true randomness, but we were able to implement one that is random enough for the purpose of our game.
- Lastly, it was also challenging to display the score. First, we attempted to draw them manually, Then, we attempted to use the texture. However, it was such a daunting task to do it for every letter and numbers, we opted for a more efficient way, which is to make use of the html webpage to display both the score and the time.

VIII. Future Improvements:

It is obvious that the gameplay experience can be enhanced with a mouse picking that controls the player’s position in the game. If only we weren’t constrained by time, for future improvements, we would implement such a feature so that the game difficulty can be reduced as opposed to playing the game using keystrokes, which can result in delay. In addition, we would also improve the pattern generation of each obstacle or even try adding more types of obstacles to the game.