**Developer’s note:**

I challenge you to play the game first before reading the following sections, they are filled with spoilers and hints. Do feel free to explore the game, and have fun. If you have any trouble, you can come back and continue reading.

**Overview:**

The animation is a puzzle-based story. It is a sinister take on the nursery rhyme “Twinkle, twinkle, little star”. It is a puzzling story full of surprises and riddles and uses dark humour. The game is mainly targeted at teenagers and young adults. It is not recommended for young children.

**Video link:** https://youtu.be/u-ZtQcwOY3U

**Time spent:** 12 hours

**No external assistance**

**References:**

bomb.png: http://simpleicon.com/bomb.html

star.jpg: https://www.kisspng.com/png-star-cliparts-69039

**Planning:**

I was interested in composing my own music, and this project provided me an opportunity to do so. As such, I chose to compose a melody based on “Twinkle, twinkle, little star” as I wanted something simple yet recognisable. Then, I wanted to make my game surprising, so I included dark humour, as that would be the last thing a person would expect from a game based on a nursery rhyme. I also wanted to include minigames and riddles to make the experience fun yet challenging for the player.

This was the final draft of the program.

*[*

Sinister twinkle twinkle little star

Twinkle twinkle little star

Happy rendition of twinkle twinkle little star plays in background

Rearrange blocks to form the next line

-4 blocks placed vertically

-click on block to switch with block n+3 th position

-"Twinkle", "twinkle,", "little", "star"

Background of little star's face

Every time you click the screen, star expands and becomes angrier

-eyebrowns dip by an angle till a limit

-expands

-becomes redder

How i wonder what you are

What am I?

A monster or a nice monster

Switch positions when hovering over nice

-3 transparent bombs in the top corner

-colors 1 bomb when clicked on monster

-when 3 bombs are activated, screen blows up into white

Up above the world so high

It's a mighty long fall down

Would be a pity if...

Like a diamond in the sky

Did you know diamonds have a melting point of...

3300°c or yes

Bet you did not know the human body is incinerated at 1500°c!

Twinkle twinkle little star switches to angry battle music

*]*

However, after programming the first iteration of the game, I realised certain features were unnecessary and removed them.

**Learning points:**

Time management:

I started on the project late into the March Holidays, as such I was quite rushed and may not have been able to deliver to the best of my abilities. In the future, I will start on projects earlier.

Planning:

I started planning the project very early, going through many drafts of the structure of the animation. In addition, each draft was very detailed, each scene storyboarded to the finest details. This made me clear on what I had to do program when I actually started on it, and whether I could achieve it. I will do the same for future projects.

Cluttered code:

The final program came out to be around 800 lines of code. I think it is very cluttered and following are a few ways I can prevent this in the future.

1. Do not write fewer lines at the expense of clarity. I often compress many lines into one line to save space, thinking it saves time and memory. In reality, and savings are minute and readability should be prized over it.
2. Variable naming conventions. I often name variables on a whim. I may be able to find those variables easily at that time, but a reader, or even myself at a later date, may not easily understand what the variables represent.
3. Separate the program into different python files. On hindsight, I should have separated certain classes into different files to cut down on clutter.

Game freezing:

The game freezes when the user tries to perform any actions while the game is paused (pygame.time.wait is called). This is because when the game is paused, the whole game grinds to a halt and the while loop accepting events does not run. In the future, I can prevent this by not using pygame.time.wait and instead look into implementing a timer that can time continuously with the while loop.

Errors:

I did not experiment fully with edge cases and different operating systems. As such, unexpected errors may arise which I did not discover and catch. In the future, I would read the documentation deeper to find out the extent of the programming features so I can better implement error catching.

**Notable features:**

Originality:

I tried to make the whole animation from scratch. I used as little third party assets as possible and tried to design my own as far as possible. However, I still used third party assets for trickier images. I also composed my own soundtrack, based off the tune of “Twinkle, twinkle, little star”. I wanted to do this as I wanted a taste into indie (independent developer) games’ developers development process.

Efficiency:

I tried to make the application streamlined and efficient. I split the main functions into clear parts. First, I drew and set-up the scene. Then, I waited for input. Then, I performed actions with that input. This was the case for all the scenes in the animation. Repeating the process made things simpler and easier to understand.

OOP:

I made heavy use of OOP (object-oriented programming). I found that OOP was often the most efficient and easiest way to program things, especially when reusing the object in different scenes, or making many copies of that object in one scene. OOP made controlling all the objects easier. The organisational structure of the classes also made programming and reading the code clearer.