My project is a very basic level of Mario. I used OpenGL and GLUT for my implementation. The game consists of Mario, Goomba, 2 pipes as obstacles and a flag to win the game. In order to be able to use double buffer and smoother transitions, I started off making an abstract Tile class. Tile class has a pure virtual draw() function, which will be utilized to draw and redraw objects, and some variables to keep track of their positions. Every other class used for the logic is a child of Tile, to be able to use polymorphism and simply be able to perform operations on all objects.

In the main.cpp file I have a vector<Tile*> which is where all the objects are stored. Then I loop through the vector and draw all of them which is the starting point of the game. Then for the rest of the game logic I utilize glutIdleFunc and glutKeyboardFunc to redraw each frame. The keyboard function listens for inputs w, a, d, which are used to move Mario up, left and right.

The Idlefunc is where most of the magic happens. The idlefunc calls all of the update functions, which check if something needs to be moved and adds to the offset variable for that object. Then it clears the buffer and redraws everything. This updates everything properly, since all the objects that would need to move at some point use an offset in their draw function. So using polymorphism and calling draw on all elements on the vectors redraws everything in their according spots.

Goomba, which is the enemy, moves autonomously. Its trapped between two pipes so it goes between the two pipes forever. If Mario interacts with Goomba, he dies and the game is over. If Mario jumps over the pipes and Goomba, he reaches the flag and he wins the game.