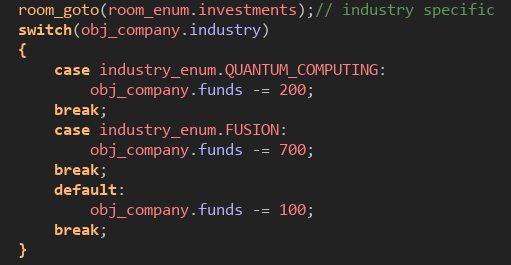
**Comments on Code:**

* oPlayer:
  + This is the object that the user controls in the grant minigame
  + The object is initialized with horizontal, vertical speeds, and gravity (hsp, vsp, grv,walksp), a bool for if they are on the floor(on\_floor), number of bags grabbed (num\_grab), and global variables for a time function (global.seconds, steps)
  + During each step of the game, the program checks
    - If the right,left, or up keys are pressed and calculates the amount the user moves
    - If the user if colliding with any of the walls in the room and stops them before they can
    - The user’s value for horizontal speed and on\_floor bool to switch between animated sprite should that are used to correspond with the motion
    - The user’s num\_grab and if it is equal to 10, it takes the user to the win page
    - The steps variable and increments it by 1. Once the steps variable equals the room speed, global.seconds is incremented by 1
* oGrab:
  + This is the object that the user is meant to “grab” in the grant minigame
  + It’s main function is to stay static and if the oPlayer object touches an oGrab object in the game, it will trigger a collision event
  + In the collision event, the num\_grab variable in oPlayer will increment and the instance of oGrab will be destroyed
* winText
  + This is the text that is displayed in the room after the user completes the grant minigame
  + A text is generated to show how well the player did based on the global.seconds variable
  + The global.seconds variable also determines the amount the game point systems (
* Fusion/Genetics/Quantum (1-3) Object
  + These show the three randomly triggered events which pass the day on their respective rooms. Each draws a text at the top showing what you accomplished and an image to accompany it.
* Game\_over\_control
  + This checks if the user enters the enter button and if so restarts the game. This is used on the game over screen in order so that the user can play again!
* generalmenu
  + This is what encapsulates all the many different options that a user may choose from the menu. Depending on what is chosen, it will “go\_to” a respective room.
* obackground
  + These are the backgrounds for each industry in the game menu once the game actually starts and are used in numerous rooms.
* obj\_Menu
  + This is what encapsulates all the many different options that a user may choose from the menu. Depending on what is chosen, it will “go\_to” a respective room.
* Obj\_bimage
  + This is the initial for the main menu (all other background gifs thereafter are dependent on the industry chosen).
* Obj\_button\_40, 55, 80
  + These are objects that are inherited from obj\_button\_parent, and allows the user to modify the number of hours that they are working.
* Obj\_button\_parent
  + This is the base class wherein we instantiate the buttons that we use for adjusting working hours in the menu of the game. All buttons (40,55,80) are children of this parent class
* obj\_company
  + This is the persistent object which tracks the most important variables in the game, the company funds and market share, as well as the industry.
  + This object also checks for the game win or lose conditions and switches the room if that happens
  + This object also contained an enumerated state for the industry in order to make checking for it and setting it throughout the game much easier to understand.
* obj\_other\_company\_2
  + This object serves as the opponent in the game and updates every time the room switches in order to guarantee that itself and obj\_other\_company\_1 comprise the rest of the market share left in the market (out of a total percentage of 100).
  + This also fluctuates by -3%-3% each cycle in order to make the competition more interesting (randomly decided)
* obj\_fundsANDmarket
  + This is the persistent object we made that stores the capital and market share progress of the individual throughout the game.
  + It will consistently update as events progress, and be displayed in the top right corner of the game.
* Obj\_industry
  + This is where the user will initialize the industry they chose. They may choose from quantum computing, genetic engineering, or
* Obj\_investments
  + This is where we store
* Obj\_learn\_more\_text
  + This controls the learn more room. This displays three different sets of text, each of which is switched between by pressing the spacebar and formatted to fit in the window. After the third spacebar it goes back to the main menu.
* obj\_market \_axis
  + This is the object that handles the axis’ of our ‘market share’ option from the menu
  + Users will access the market share to see how they are doing relevant to competition (where competitors' market share is modeled by: 1/Marketshareours
* Obj\_market\_graph
  + This is where we drew the graph for the market share. Essentially, all the bars tracking market share progress is a rectangle that is growing with successive modifications of stats (the bar width is 143 pixels) and the max “market share” is really just a pixel size of 495.
* Obj\_person\_typing
  + This is the object we made for a sprite we designed simulate the typing on a keyboard in our waiting room
  + Users will access the waiting room from the list of options throughout the game
* Obj\_research
  + This is where, depending on the industry chosen by the user, the buttons in a menu will update and the user will be able to select from two different options to how they want to invest
  + This is all encapsulated within a switch statement that, depending on the research they choose, will alter the global member variables (market\_share and capital)

Example of incrementing based on industry

* Obj\_rooms
  + This persistent object creates the enum for objects which switches between the rooms. This allows us to easily reorder the rooms without affecting in game functionality. While not necessary it was good style and very useful. The mapping is as follows:

enum room\_enum{

initial = 0,

main\_menu = 1,

learn = 2,

industry\_perspective = 3,

industry\_choice = 4,

research = 5,

investments = 6,

game\_menu = 7,

working\_room = 8,

adjust\_working\_hours = 9,

vacation = 10,

grant\_minigame\_0 = 11,

grant\_minigame\_1 = 12,

market\_shares = 13,

conference = 14,

game\_over = 15,

fusion1 = 16,

fusion2 = 17,

fusion3 = 18,

genetics1 = 19,

genetics2 = 20,

genetics3 = 21,

quantum1 = 22,

quantum2 = 23,

quantum3 = 24

}

* Obj\_vacation\_choice
  + This is the object that holds all of the vacation choices. Depending on the randomizer, the user may experience different vacation screens (or none at all -- with a witty jeff bezos quote)
* Obj\_working\_hours\_title
  + Shows the text "How many hours should they work?" in the adjust working hours room
* Obj\_working\_text
  + A deceivingly important object this handles all of the working room functionality. This means that it decrements the funds and increases the market share for each room relative to the industry. This is accomplished with the persistent member variables obj\_company.market\_share\_per\_day and obj\_company.funds\_gained\_per\_day which are set in the adjust working hours room.