FOR TEACHERS EYES ONLY!!!

Tech Fair CSI Project

- Create a game in the time allotted (60 minutes)
- Your game must have 3 different sprites (1 to control, 1 to collect, 1 to avoid)
- Your game must have at least 3 different backgrounds
- Your game must have a beginning with instructions and an end (you win or you lose)
- Your game must have 2 different variables (health/life, score)
- The draw loop should only contain functions and the draw sprites command
- Variables should be created above the draw loop
- Functions should be created below the draw loop
- See the rubric below for how the judges will be assessing your game
- ## Rubric

Key Concept	Extensive Evidence	Convincing Evidence	Limited Evidence	No Evidence
Program Development - Program Readability	Your program code effectively uses good naming conventions for sprites, non-sprite variables, and functions, and additional comments are added throughout the program to make the code easily readable.	Your program code makes use of additional comments throughout the program.	Your program code has few comments beyond starter code.	Your program code does not contain comments and is difficult to read.
Program Development - Program Sequence	You *sequenced the program well*[1] and properly separated *code in and out of the draw loop*[2].	Your program effectively creates sprites and defines functions outside of the draw loop, however there are multiple code segments within the draw loop that should be situated within their own functions outside the draw loop.	You have significant sequencing errors, which include either sprites created within the draw loop or functions created within the draw loop.	Your entire program code is within the draw loop.
Modularity - Use of Functions	At least three functions are created outside the draw loop and used to organize your code into logical segments. At least one of these functions is called multiple times in your program.	At least two functions are created outside the draw loop and used in your program to organize your code into logical segments.	At least one function is created outside the draw loop and used in your program.	There are no functions created outside the draw loop and used in your program.
Algorithms and Control - Backgrounds and Variables	Your game has at least *three backgrounds that are displayed during run time*[3]. At least one of these *backgrounds is triggered automatically through a variable*[4].	Your game has at least two backgrounds that are displayed during run time.	Your program code has multiple backgrounds but only one is ever displayed during run time.	Your game has one background.
Algorithms and Control - User Input	Your program responds to at least 2 *different types of user input*[5].	Your program responds to 1 type of user input.	Your program contains code for detecting user input, such as keyDown(), but it is not used correctly and the program does not respond.	Your program does not use any code for detecting user input, such as keyDown() or mouseDown().
Algorithms and Control - Interaction Conditionals	Your program includes multiple conditionals to *control complex interactions between sprites*[6] which either affect the sprites themselves, such as resetting	Your program has multiple conditionals meant to control complex sprite interactions, however one of them does not work as expected.	Your program either has only one conditional to control complex sprite interactions, or has multiple conditionals but none of	Your program does not use any conditionals to control sprite interactions.

	their location, or affect the game play, such as increasing or decreasing the score.		them work as expected.	
Position and Movement	Your program includes at least two *instances of complex movement*[7].	Your program includes at least one instance of complex	Your program includes *simple, independent movement*[8].	There is no movement in your program, other than direct user control.
Variables	Your program properly creates at least 2 non-sprite variables, such as score or health, that are displayed and updated during the game and *affect how the game is played*[9].	Your program properly creates at least 1 non-sprite variable that is displayed and updated during the game and affects the way the game is played.	Your program properly creates at least 1 non-sprite variable and updates it during the game but it either is not displayed or does not affect the way the game is played.	Your program does not create or update any non-sprite variables.

1. Sequenced the program well

- If the program code is not sequenced correctly, some elements, such as shapes, sprites, and text, may be unintentionally hidden behind others.
- In the specific case of sprites, if the `drawSprites()` code is not sequenced correctly, some or all sprites may not appear on the screen.

2. Code in and out of the draw loop

- Code outside the draw loop is used to set up the program, its starting elements, and all created functions.
- Code inside the draw loop at this point should be primarily function calls and perhaps some conditionals used with function calls in addition to the `drawSprites()` block.

3. Three backgrounds that are displayed during run time

- Three different backgrounds should appear during a single play of the game.
- Example backgrounds include (but are not limited to): an instructions screen, different backgrounds for various difficulty levels, win screen, lose screen, etc.

4. Backgrounds is triggered automatically through a variable

- For example, the game play background is changed to a "*You lose*" background when the variable named "score" reaches the value of 0.

5. Different types of user input

- This includes keyboard and mouse inputs.
- Different types can include two different keyboard inputs such as `keyDown("left")` and `keyDown("space")`

6. Complex interactions between sprites

- The use of a single collision block, such as `collide()` or `displace()`, without the use of a conditional does not count as complex interactions

7. Instances of complex movement

- This includes code that simulates movement such as acceleration, moving in a curve, looping from one side of the screen to the other, or jumping

8. Simple, independent movement

- This includes movement such as moving in a straight line or rotation

9. Affect how the game is played

- Examples include (but are not limited to): moving to a different level at a certain point value, losing the game at a certain score value, game play becomes harder or speeds up at a certain variable value, etc.