Name ____ Period ____ Date________

Project 3: Advanced platform game

Students will implement a side scrolling platform game, example Super Mario Bros.

Overview

Platform games are among the most widely recognized types of video games. Composing about one third of all console games at the peak of their popularity, platform games are characterized by their relative simplicity and by the common game play element of jumping across suspended platforms (hence the name) to avoid falling into a hazard. Platform games also typically include enemy characters, items that grant the hero special abilities ("power*ups"), and a "checkpoint" system that allows the player to restart from partway through a game or level when he or she dies.

Big ideas

Personal design interests require the evaluation and refinement of skills; Tools and technologies can be adapted for specific purposes

Consider using the familiar platform game concept to not only entertain, but also educate or bring awareness to social concerns, an environmental issue, or some other context.

Example 1: In Pacific Northwest, there are many efforts exist to promote and guide a sustainable future of wild Pacific Salmon and their habitat. Students can create a game where the player can help salmon swim back to the stream where they were born, jumping upstream through ladders and fishways, battling hazards and enemies (predators, fishermen). Game play can introduce elements of education and awareness.

Example 2: The West Coast has been battling with more Wildfires than ever in history. Students can create a game where a traveler drives through different towns, parks, and highways in your area, extinguishing hazards (cigarette butts, campfires), battling enemies (fires out of control, irresponsible tourist), enforcing fire bans, reporting fires, being a good neighbor, implementing home prevention, rescuing animals, and so on. Game play can introduce elements of geography (parks and forests, lakes and waters) and climate and terrain (temperature, animal habitats). Read more about wildfires in here:

How can the classic platform game be adapted to a specific purpose, cause, or context that interests you, or impacts your community?

Behavior

Game play

In a platform game, the player controls a hero who moves throughout the world attempting to reach an endpoint and/or accomplish a goal. Along the way, the hero encounters hazards such as pits (into which he or she can fall) and enemies (which can either move or be stationary). The hero has a finite number of chances (known as “lives”) to achieve his or her objective. Each time the hero succumbs to a hazard, a life is lost and the player must try again.

The hero

The hero moves around the world under the player’s control. The hero can perform three basic actions: move left and right (controlled by the arrow keys) and jump (by pressing the space bar). As the hero moves throughout the world, he or she is subject to gravity. This means that when the hero jumps or moves off the edge of a platform, he or she should return quickly to the ground (or another platform). The hero should never fall through a platform or the ground.

Screens

Your platform game will be a side*scrolling game. In this style of game, the scenery changes as the player moves horizontally across the screen. While many modern side*scrollers (including Super Mario Bros.) scroll smoothly as the player moves, our game will use a simpler system and consist of screens. Each screen represents one section of the overall world in which the game takes place. The hero should be able to move freely around each screen, but when the hero reaches the far right edge of a screen, the next screen should appear and the hero should be placed on the far left edge at the same height. Your game must include three distinct screens, and each screen must have one hazard.

Hazards and lives

Your script should include one of each of the following types of hazards:

- A falling hazard (a hole, pit, or other opening) into which the hero can fall if he or she does not jump to avoid it. Falling to the bottom of this hazard causes the hero to die.
- A stationary enemy that does not move, but that causes the hero to die if it is touched.
- A moving enemy that also causes the hero to die if it is touched, but that moves in some way. Note that an enemy can be either a character (like Goombas in Super Mario Bros.) or an environmental hazard (such as spikes). When the hero dies by either falling down a falling hazard or touching an enemy, he or she loses a life. If the hero has lives remaining, one should be lost, used power*ups and defeated enemies should be reinstated, and the hero should be placed back at the left edge of the current screen. Otherwise, the game is over and a suitable message should be displayed.

Power Ups

While moving through the world, the hero may obtain power*ups that grant him or her special abilities. Examples can include increased jumping power, invulnerability, or the ability to destroy enemies. These abilities should be temporary, and there should be some visual indication when the hero has access to them. A power*up should appear as an item (sprite) in the world. The hero will obtain the abilities by touching the power*up sprite, at which point the power*up should disappear. Your game should include two distinct power*ups, one of which is required for the hero to win the game. In addition, one of your power*ups should be hidden, meaning that the player must take some action before he or she can obtain its abilities. For example, in Super Mario Bros., Mario must jump into a special block to make many power*ups appear. A hidden power*up should not be visible until it is triggered by the hero, and the hero should not be able to obtain the abilities until the power*up is revealed.

Winning the game

There should be some clear end goal that, when achieved by the hero, ends the game in victory. This victory condition should be obvious even to a new player. You need not (and probably should not) provide written instructions to the player about this condition. Make sure to use easily identifiable visual indicators such as flags, doors, etc.

Reset button

At any point during game play, if the player presses the ‘z’ key, the game should reset to its initial state. The game state after pressing the ‘z’ key should be indistinguishable from when the game first begins. This means, among other things, that:

- the hero should return to the left edge of the first screen, lose all special abilities, and be given back all his or her lives
- any destroyed enemies should be reinstated and replaced in their original positions
- obtained power*ups should become available and revealed power*ups should be hidden

Implementation details

Design and creativity

Your script should be well designed and have a unifying theme, characteristic, or style. This can be a particular style of artwork, common colors, and/or related types of characters. In addition, you should show some effort and creativity in your design. Do not simply recreate an existing game or use only ideas put forth in this spec. Come up with some original concepts for characters, backgrounds, power ups, etc. and utilize them in your game. If you make us say “Wow!” it may even be worth some extra credit. Using copyrighted assets (including characters or artwork from an existing game), even with modification, is not allowed.

Custom blocks

Throughout your script, you should use custom blocks to generalize common operations and increase the readability and maintainability of your script. Your script must include at least three custom blocks, at least one of which must take arguments. Do not limit yourself to just three blocks: use custom blocks (including arguments and reporters) anywhere you feel it will help your script. Part of your grade will be based on not only meeting the minimum usage requirement but also on your decisions of when, where, and how to use custom blocks.

Documentation

In addition to functioning well, your script must be well documented and readable. This includes, but is not limited to, things such as:

- organizing your scripts so that they can be read and comprehended easily
- giving your sprites meaningful names
- naming and using your variables well
- including comments to describe the structure of your script and any particularly complex or unintuitive pieces of script

Required elements

Your script must include, at a minimum, the following script elements:

- At least two variables
- At least one conditional (if or if-else) statement
- At least two messages, one of which must be received and responded to by multiple sprites

Peer feedback

As part of the software development experience on this project, you will participate in a peer review with one or more of your classmates. Near the end of the project, you will play another student's game and provide him or her with notes and comments. Your partner(s) will also play your game and offer the same feedback. Describe to your partner what parts of the game you liked. You should offer suggestions for features that could be improved or changed as well as look for bugs in the script you are reviewing. Keep your comments constructive and professional! Don't just point out things you don't like—explain your thinking and propose solutions. Also, restrict your comments to things that can be reasonably addressed. Do not tell your partner that he or she made a poor choice of theme and should start over, for example. After receiving your peer feedback, you should consider the comments carefully and respond. You will be expected to turn in the feedback provided to you and identify ways in which you modified your game in response to the feedback you received.

Curriculum competencies - design sharing

As you create software, you will need to put yourself in the end user's position, and imagine yourself using the software. At the same time, providing objective, critical analysis and feedback of other people's work will not only benefit others, but also help you in your own design work. Being able to welcome, and provide, feedback are essential skills for a software developer.

Required checkpoints

1. Screens should be designed; the hero should be able to move and jump; gravity should work; reset button should be functional
2. Hazards and enemies should be present; death should work properly.
3. Lives, power*ups, and victory should be implemented; all other required script components must work

Grading rubric

Functional correctness (behavior)	**Points**
Left arrow, right arrow and space bar control hero's movement. Hero does not move through walls.	2
Hero is subject to gravity, and does not fall through platforms.	2
Game consists of three screens.	1
Game contains One falling hazard (pit), one stationary enemy, and one moving enemy.	3
Player loses a life when falling down the falling hazard or touching an enemy. Hero restarts on current screen after death.	2
Player starts with three lives and game ends when player is out of lives.	2
Game contains two power-ups, one of which is hidden and one of which is necessary to win the game.	3
Hero has a clear goal to win the game.	1
Game play is clear and intuitive, even to a brand new player.	1
Game resets when the 'z' key is pressed.	2
Total	**19**
Technical Correctness (Implementation)	
Well designed visually and has a consistent theme.	2
Shows good creativity and effort.	2
Well documented and exhibits good style.	2
Includes at least two variables.	2
Includes at least two messages, at least one of which is received and reacted to by multiple sprites.	2
Includes at least one conditional statement.	1
Includes at least three custom blocks, at least one with arguments.	4
Custom blocks are used where appropriate.	2
Provide valuable test feedback to at least two other students.	2
Test feedback from at least two other students.	2
Checkpoint 1	**2**
Checkpoint 2	**4**
Total	**27**
Grand total	**46**