SpaceRace

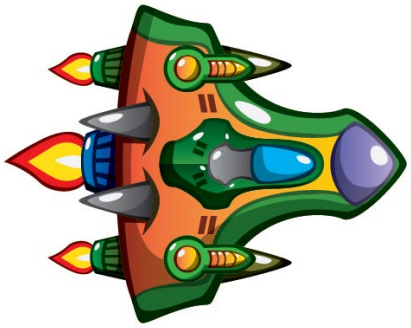
Kenny Cox 20549345/BU

Table of Contents

SpaceRace

Intro

SpaceRace is a top down, one player, easy fun space (futuristic) race game. The object of the game is to collect "clocks" to add time in order to complete a level. Clocks appear and disappear randomly on the race track(s) every 3-10 seconds. If the player fails to collect enough clocks to gain additional time or fails to complete the required amount of laps, the player will lose and have the option to play again or quit the game. Score is tallied at the end of each game. The players score will be saved on a leader board, only top 3 three scores will be saved. Fun easy game for kids between the ages of 12-16 will enjoy this space race game.



Theme

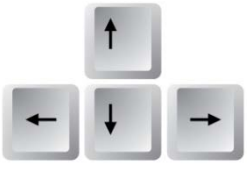
The intended theme is a futuristic space race, the year is 2027, the driver is a 21 year old alien named Kruff from the planet Vernolium. Every year his planet holds an annual space race event. Kruff has been training since he was allowed to drive. Kruff has to race his way to the top of the score board by collecting as many coins as possible while staying on the track and avoiding obstacles.

Boller (online), "A game can have a theme but no story, a theme and a story, or no story and no theme (think Scrabble)."

The theme will affect the key elements (track, car). Futuristic cars flying through the track(s), exploding cars, floating stands for the spectators surrounding the track. The spectators will be alien like and in the background of the race. A regular Nascar race car would not suffice for this particular theme. S. Boller (online), "**Knowledge Guru** uses thematic elements to convey the idea of theme even though there is no real story." Icons (i.e. coins) will have glowing effects as to look like a space object. Objects in the game will need to be consistent throughout the game. For a better idea of the games properties, please view the storyboard designs. atartiarchives.org, software & info (no date) stats that, "Goals must suit a player's expectations or fantasies." By allowing the player to have a goal, collect coins for points and for much needed added time in order to complete the game and scoring top points, gives the player a sense of purpose.

Controls

The player can control the ship by utilizing the arrow keys, details are in the chart below. In flash the 0 rotation starts facing right, 90 faces down, 180 faces left, and 270 faces up (clock-wise).

UP ARROW	FORWARD	
DOWN ARROW	REVERSE	
LEFT + RIGHT ARROWS	STEER	

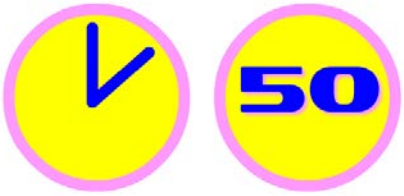
Sounds

Sound will give feedback for the following objects. Sound effects will occur when the player collides with coins, access the navigational buttons/panels etc. A spacey background music will be placed on stage, each scene will have a similar sound. A thump sound occurs when the vehicle hits a barrier. Explosion sound effect for when the car explodes upon running out of time. Vehicle effects will be required, such as engine starting, tires screeching from braking hard, or accelerating quickly. Sound is very important (along with supporting animations, i.e. twinkles) to relay information back to the player, it sets the mood, so it is important that the fits the theme, futuristic and in return, it will create more realism and overall better gaming experience. The sound used within the game has to be royalty free music, such as the following three sites, FreeSound (https://www.freesound.org/),SoundBible (http://soundbible.com) and FreeFX.com (http://www.freesfx.co.uk).

Objects

Coins

an animation occurs and a twinkle sound occurs simultaneously allowing the player to know that they hit the object. clock or coin animates in an upward direction after colliding with the players car thus adding time/points to the players score/time or award system.



1. clock coin - adds time to the games clock, the games clock counts down from 1 minute. Coin time values in the game are 10 and 20 seconds respectively.

2. points icon are exactly that, points. Points will be awarded to the player once they are collected. Denominations of 10, 20 and 100 points are awarded.

Obstacles

Boundaries can be found throughout the track and only hinders the players way. no damage is caused. the ship bumps off, makes a "thump" sound. Please view storyboard for a better idea of how the obstacles look and perform.

Algorithms

The track will be added by using a Movie Clip and removed when the game is over. Other Movie Clips will be used for the other screens/scenes of the game.

Imports : import flash.display.MovieClip; Imports allow us to take the code from another class and use it in ours.

User input : Since each key on the keyboard has its own key code, I need to create an event listener within a function to listen out for when the player holds a key down. If the key is pressed, do something, i.e. accelerate/reverse.

Vehicle will be a movie Clip and it will be called onto the stage when the game starts.

Vehicle acceleration : a stage event listener listens out for each time a button is pressed (to move vehicle), the speed increments by a set value, simulating acceleration. This needs to be created by assigning an integer value to a variable, 0 is the default number given by Flash. When the car hits a barrier, the speed will be affected.

Vehicle deceleration : The car loses speed over time when the acceleration button isn't pressed, giving it a more realistic feel to the game. .5 will be removed from the speed when the player removes his finger from the up arrow for every second/half a second. Testing is required to get a better idea of what looks better.

Vehicle top speed variable is created and assigned an integer value, for example a value of 10, which would be considered the vehicles max speed.

Turning/Velocity : When UP arrow key has been pressed the y velocity needs to be adjusted based on the direction of the ship. **vy += Math.sin(degreesToRadians(rotation)) * speed;** can be used, for example, If the rotation is 0 or 180 Math.sin will return 0 so the vy isn't affected. But if the rotation is 90 or 270 it will return 1 or -1 and therefore fully affected. The same is done with the vx value (except with cosine) to get our results in x. Adding realism to the game as the game.

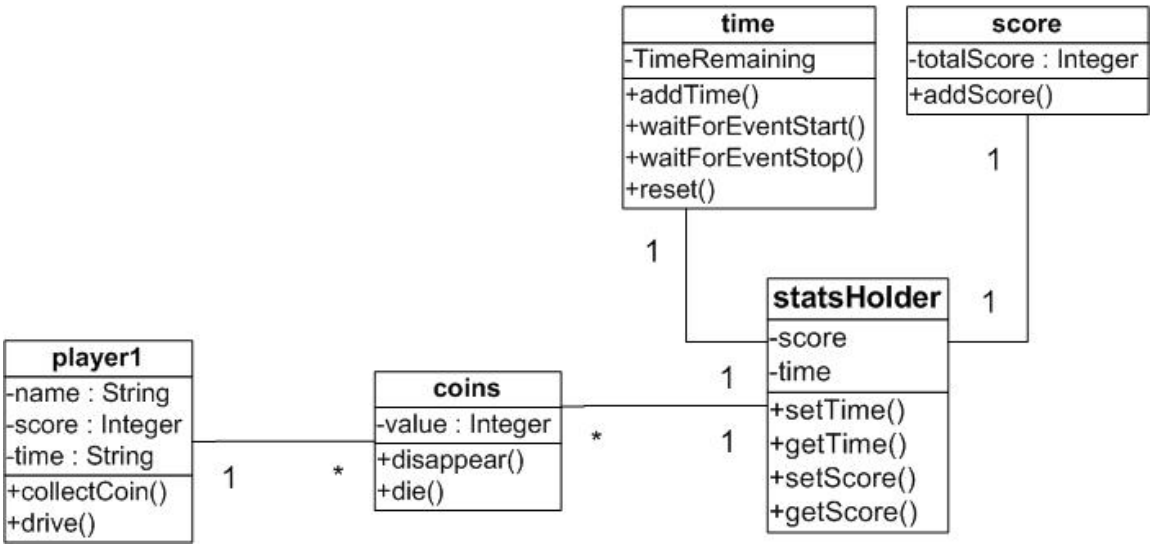
Collision detection is important and HitTest is one of the built in function available in Flash actionscript to make the DisplayObject to interact with other objects. The hitTestObject function returns a Boolean variable of either true or false depending on whether or not the two objects are hitting one another. Alternatively, I can use the hitTestPoint() method, which is to check the overlapping of given point with display object. For example, I can use the hitTestObject for the barriers and invisible barriers, and hitTestPoint for the coins, since the coins are round, it would be more a more realistic and accurate depiction of real life. If statements are used to detect collisions. syntax Object1.hitTestObject(Object2);

Gravity : Since the car can fall off the track, gravity will be required, a stage event listener is created to check falls off the track. if this happens, the gravity should allow the vehicle to fall into space. Gravity is simulated on the race track, hence the reason why the cars do not float away while in space. keeping in mind that this is a top down game, the car would only get smaller when falling.

Polymorphism

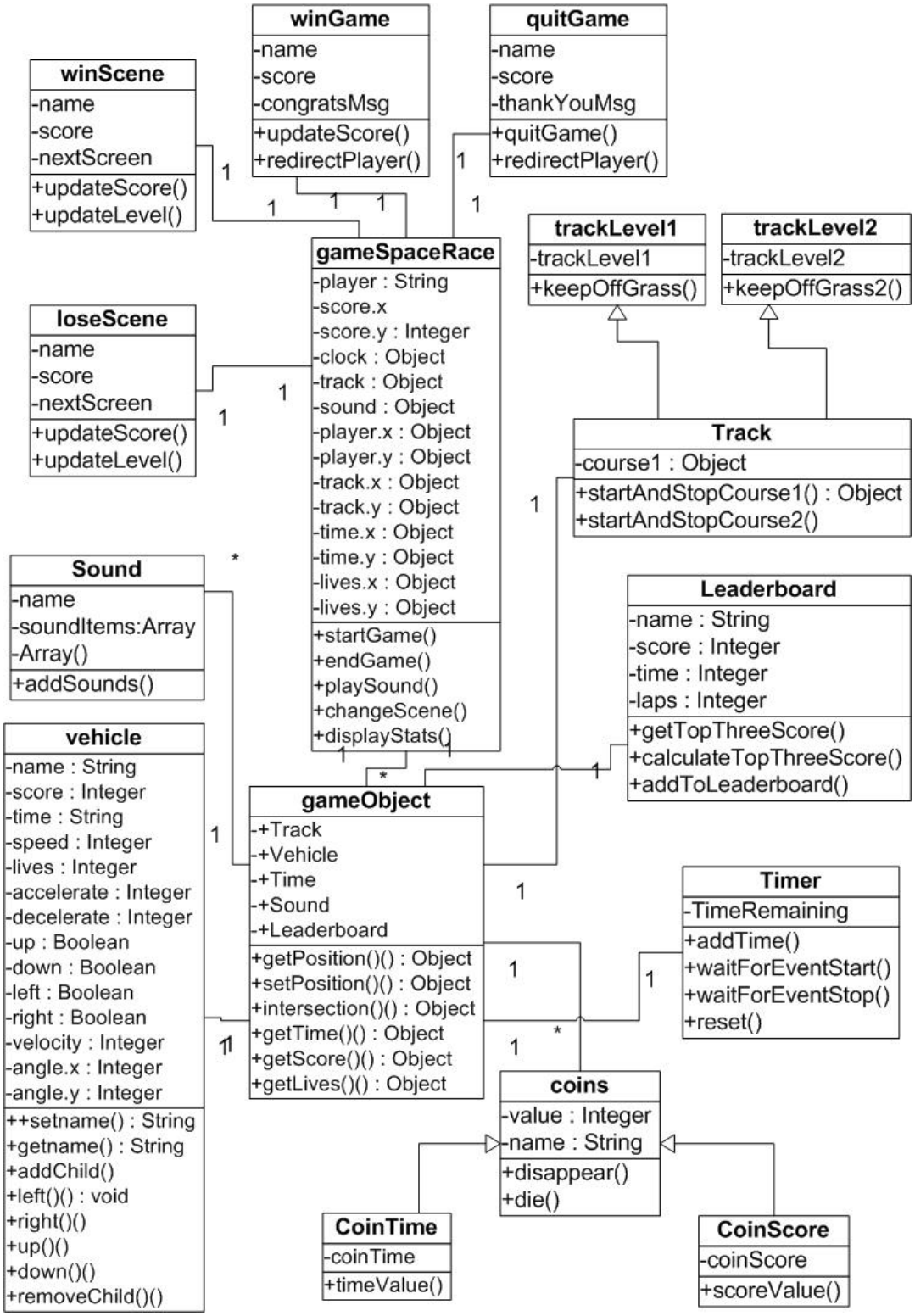
Adobe (2012) "Polymorphism is the concept that multiple types of objects might be able to work in a given situation." For example when the player object collides with the clock object, the score/time object(s) update accordingly. The diagram below is a Polymorphism example in my game.

The Player collides with coin, the stats holder updates either time or score accordingly.



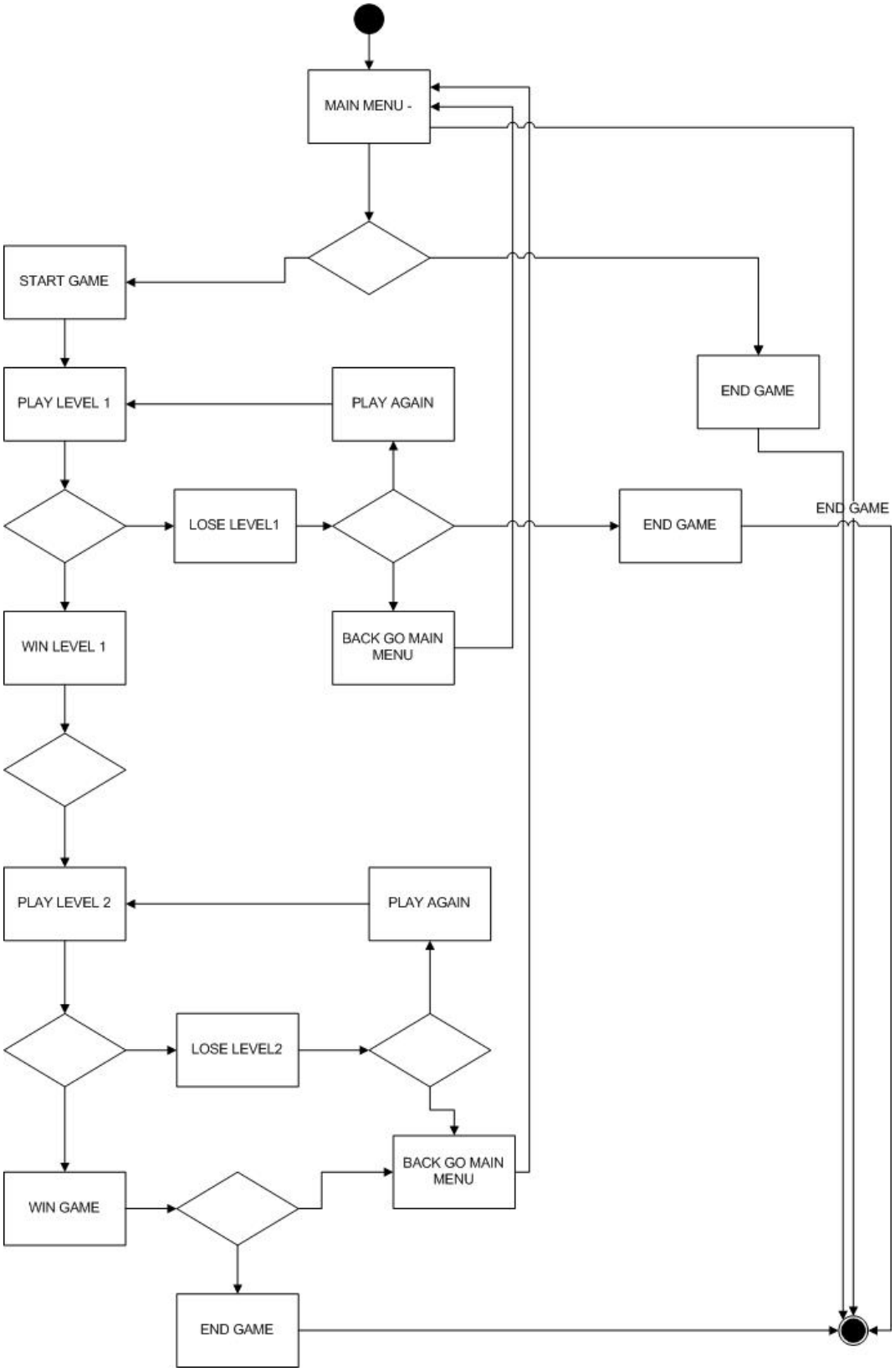
Class diagram for SpaceRace.

Adobe (2012) "Inheritance enables new classes to receive—or *inherit*—the properties and methods of existing classes." In the game I have two types of coins, one coin adds time and the other coin adds points to the score. both coins inherit the parent class coin. Also, two track levels inherit the parent class Course/track.



Activity Diagram

Player starts the game and is directed to the main menu. The player has the choice to start game or end game. if the player ends the game, the game ends. if the player decides to play the game, the player is directed to level 1. The player can either win or lose the level. if the player loses the level, the player can either quit or try again or go back to the main menu. if the player decides to continue, the player is directed to level two. Again, the player can either win or lose the level. if the player wins the level, the player wins the game. if the player loses the level, the player can try again, quit or return to the main menu.



References

- M. Yaiser, Feb 6, 2012., Adobe, *"Object-oriented programming concepts: Polymorphism"*, http://www.adobe.com/devnet/actionscript/learning/oop-concepts/polymorphism-and-interfaces.html
  [accessed on Nov 14, 2014]

- Teacher Websites © 2014 Edline, Thama Valley District School Board. *"Kenji Takahashi's Resource Site"*.
  http://www.tvdsb.ca/webpages/takahashid/video.cfm?subpage=128730
  [accessed on Nov 19, 2014]

- S. Boller, no date, *"Learning Game Design Series, Part 5: More Game Elements to Consider"*, http://www.theknowledgeguru.com/learning-game-design-series-part-5-aesthetics-theme-story-resources/
  [accessed on Nov 17, 2014]

- atartiarchives.org, software & info (no date), *"Game Design Theory"*, http://www.atariarchives.org/agagd/chapter10.php
  [accessed on Nov 15, 2014]