CSCI 103 Introduction to Computer Game Development  
Winter 2016  
Laboratory Exercise 4

Objectives

1. Practice with the use of paths, tiles and alarms.
2. Drawing the score within the game window.

Submitting Your Work

Save your game file as $\text{Wnnn}\text{nnnnLab4.gm81}$, where $\text{Wnnn}\text{nnnn}$ is your WWU W-number, and submit the file via the Lab Exercise 4 Submission item on the canvas web site before the deadline.

Exercise Steps

In this exercise, you will develop a game that involves a car chase through city streets. The player drives a red sports car that is being chased through the city streets by several police cars. As it drives through the city streets, the red sports car is collecting bundles of cash left lying on the road.

1. The Game Elements

For this game, you will need the following resources, all available on the course web site:

a. Sprites:
   - Four animated sprites for the police car, each facing one of the four directions.
   - One sprite for the crashed police car, with 4 images corresponding to the crashed police car facing each of the four directions.
   - Four sprites for the sports car, each facing one of the four directions.
   - A single-image sprite for the money.

   Center the origin of all these sprites.

b. Tiled background BigStreets.bmp, containing tiles 64 x 64 pixels, with 2-pixel separation.
2. City Streets

a. Create a background using BigStreets.bmp and set it up for use as a tile set. To do this:

i. In the Background Properties window, check the box “Use as tile set” just above the OK button. This causes the Tile Properties information to appear in the Background Properties window.

ii. In the tile properties, set the tile width and height to 64, set the horizontal and vertical offsets to 0 and set the horizontal and vertical separations to 2.

![Background Properties window](image)

b. Create a room, with grid snap values of 64, and set the room size to 1024 x 704.

c. Layout the streets of the city using the tile set. This is your big opportunity to try some town planning!

d. Now we create an object to keep cars on our roads. Use the sprite editor to create a 16 x 16 pixel transparent sprite with an X marked through it, to mark the roadside borders. Uncheck precise collision checking for the sprite. Create an invisible, solid object with this sprite.

e. Change the room’s grid snap value to 16 x 16 and add the roadside objects along the sides of all the roads. Be careful how you place these objects on the street corners. If they protrude onto the street, the cars will get stuck on them.
3. Police Patrol Routes

a. Choose one point in the room where the police cars will start and finish their patrol routes (the police station). Typically, a crossroad junction is ideal for this. You don't need to put anything there, just decide where it is for the next step.

b. Add a path to the game, using the add path button or use the Add path option from the Add menu. This opens the Path Properties window.

c. In the Path Properties window, click on the room icon in the toolbar. This causes the room, its objects and tiles to be displayed in the path drawing area.

d. Start your path at the “police station” position that you chose back in step 3a. Click on other points in the path, following the city streets, until you eventually return to the police station point.
e. Repeat steps 3b through 3d to create 3 more paths, taking different routes through the city streets, starting and finishing at the “police station” position.

4. Police cars and dispatcher

a. Create a police car object and give it one of the police car sprites – it doesn’t matter which one as we will fix that later, depending on which direction the police car is moving.

b. Create a crashed police car object and give it the crashed police car sprite – it doesn’t matter which image, as we will fix that later.

c. Create a police dispatcher object, which has no sprite. Every 5 seconds this object must create a police car object at the police station, provided there are less than 5 police car instances already in existence.
   i. Add a police dispatcher object to the room at the position that you chose for the police station.
   ii. In the police dispatcher Create event, create an instance of the police car object at position (0, 0) relative and set Alarm 0 for 150 steps.
   iii. In the police dispatcher Alarm 0 event, set Alarm 0 for 150 steps and, if there are fewer than 5 instances of the police car object in the room, create a police car object at the police station position.
   iv. Add a police dispatcher object to the room.

d. We want the police car objects to make a random choice between the 4 patrol routes. In the police car's Create event:
   i. First we need to make a random choice between the numbers 0, 1, 2 and 3. To do this, set the value of variable PathChoice to choose(0, 1, 2, 3).
   ii. Now we need to set the path based on the value of the variable PathChoice. Test whether PathChoice is equal to 0 and if it is, set the path to path0.
   iii. If PathChoice is equal to 1, set the path to path1.
   iv. If PathChoice is equal to 2, set the path to path2.
   v. If PathChoice is equal to 3, set the path to path3.
If your paths have names different to path0, path1, path2, path3, substitute your path names.

vi. In setting the path, set the police car’s speed to 4, set it’s “at end” option to “stop” and its “relative” setting to “absolute”.

e. At each step, the police car must have the right sprite for the direction it is traveling. We can do this by adding a Step event for the police car and in the action list, change the sprite according to the police car's direction of motion. If direction is 0, change to the sprite facing right; if direction is 90, change to the sprite facing up; if the direction is 180, change to the sprite facing left; if the direction is 270, change to the sprite facing down.
In case you were wondering …

The step event actions described above prevent the police car sprite from showing its animation of flashing lights, since we change sprites every step, restarting the animation sequence. You don't need to fix this problem for the lab exercise but, in case you are curious, the way to fix it is described as one of the extensions to the lab exercise at the end of this document.

f. When the police car reaches the end of its path, end that path and make another random selection from among the 4 available paths, using the same set of actions used in the police car's Create event, in step 4d.

g. If a police car crashes into another police car or a crashed police car, it must:

i. End its path,

ii. Use the Change Instance action to replace the police car object by a crashed police car object in its place, choosing “yes” for the “perform events” option. This will make the crashed police car perform the actions for its Create event.

h. In the crashed police car object’s Create event, use the Change Sprite action to choose the right image for the crashed police car, so that it faces in the same direction as the police car object that crashed. Fortunately, when we changed the instance from the police car to the crashed police car in step 4gii, the value of the direction variable was preserved. Therefore we can use direction/90 as the image index for the crashed police car sprite.
i. It takes 60 to 180 steps to clean up crashed police cars. In the crashed police car object’s Create event, set Alarm 0 to 60 + random(120).

j. In the crashed police car object’s Alarm 0 event, destroy the object.

5. Sports Car

a. The sports car can be moved by arrow key presses, moving at a speed of 8 pixels per second. Use the arrow Keypress events for the Left, Right, Up and Down arrows to set the car in motion in the corresponding direction.

b. Add arrow Key release events to set its speed to 0 and snap to a 16 x 16 grid.

c. Add a Step event to choose the correct sprite for the sports car, based on its direction.

d. Add a collision event for the sports car with the roadside barrier objects. In the action list, bounce off the barrier, set the speed to 0, and snap to a 16 x 16 grid.

e. If the sports car collides with a police car or a crashed police car, the game is over. Use the Check Question action from the control actions tab to display the message: “Game over! Want to try again?”

If the player answers “Yes”, restart the game, otherwise end the game.
6. Leaving Bundles of Money Lying About

Bundles of money will be left on the street at random times, in random positions. The player scores by collecting money. We will place some invisible objects at various places on the road and, at regular intervals have a chance of creating a bundle of money at that position, if there isn't one there already.

a. Create a money object and give it the money sprite.

b. Create a “money marker” object. It does not need a sprite.

c. In the money marker object’s Create event, with a 1 in 10 chance, create a money object at position 0, 0 relative.

d. Also in the money marker's Create event, set Alarm 0 for 5 seconds (150 steps).

e. In the Alarm 0 event, if there is not a money object at position 0, 0 relative, with a 1 in 10 chance create a money object at that position. Then set Alarm 0 for 150 steps.

f. Place 10 to 20 money marker objects at various points on the streets. Since this object does not have a sprite, it will show as the little blue disk with a question mark in the Room Properties window.

g. Run the game to confirm that the money bundles appear on the road with apparent randomness.

7. Collecting Money

The bundles of money are collected by the red sports car driving over them. The player scores by collecting money.

a. For the red sports car object, add a collision event with the money object. In that event, destroy the money object and add 1 to the score.

8. Drawing the Score

Since the dispatcher object does not have a sprite, we can use that object to draw the score in the game window.

a. Add a font to the game and choose a font with size 20 or thereabouts, so the score is easy to see.
b. Choose an empty piece of real estate in the room, away from the roads, with at least 100 pixels width and 20 pixels height.

c. In the dispatcher object’s Draw event:

   i. Set the color for drawing the score. Choose a color that will contrast well with the background.

   ii. Set the font to the font added in step 8a.

   iii. Use the Draw Score action, specifying as its position the top-left corner of the area where the score is to be drawn.

   

   d. Run the game and confirm that the score is drawn where you expect it to be.
9. Randomizing the Game

Once you have tested all aspects of the game, you should make it less predictable by randomizing the random number generator seed.

a. In the dispatcher object’s Create event add the Execute Code action and In the script editor window, type the following line and then click the check-box to close the window:

   randomize();

This is the end of the lab exercise. Don’t forget to submit your game file!

10. Other Features

Other things you can try, to enhance the game:

b. Make the police car sprite animation work properly, even with the possible change of sprite at every step (described in step 4e). A way to handle this problem is to change sprites only when the police car changes direction. You can do this by:

   • In the police car Create event set the value of a variable old_direction to -1.

   • In the police car Step event, check whether the variable direction is not equal to old_direction. If it is not equal, set old_direction to the value of direction and change the sprite, as described above.

c. Keep track of the sports car’s fuel level, decreasing fuel supply whenever it moves, and set up gas stations where it can buy fuel (with the money found on the road). If the sports car runs out of fuel, it’s game over!

d. Step 5c, where we choose the sports car sprite based on its direction, could be made much easier by combining the 4 images into one sprite. Then we can choose the correct image in the same way that this was done for the crashed police car.