Managing Game States

Switch Statements and Functions

Game Programming Foundations

Last modified 08/01/15 by Sam Cartwright

Topics

- Switch Statements
- Functions
- Game States
- Tips
- Added a Splash Screen and Game Over Screen

The Switch Statement

- Select one of many blocks of code to execute
- An alternative to multiple if-else statements

Switch Statement Syntax

- The switch expression is evaluated once.
- The value of the expression is compared with the values of each case.
- If there is a match, the associated block of code is executed

```
switch(expression) {
    case n:
        code block
        break;
    case n:
        code block
        break;
    default:
        default code block
}
```

Switch Example

- Execution of the matched code block stops when the break is reached
- If there is no break, the next code block also executes
- default specifies code to run if there is no match

```
switch ( new Date().getDay() )  {
    case 0:
        day = "Sunday";
        break;
    case 1:
        day = "Monday";
        break;
    case 2:
        day = "Tuesday";
        break;
    case 3:
        day = "Wednesday";
        break;
    case 4:
        day = "Thursday";
        break;
    case 5:
        day = "Friday";
        break;
    case 6:
        day = "Saturday";
        break;
    default:
        day = "OMG The Universe is Imploding!";
        break;
}
document.writeln(day);
```

Use a Switch Statement When…

- You have more than 3~4 if-then statements
- You think you might add more cases later

Bad

```
if(number == 0) {
    doStuff();
}
else if(number == 1) {
    doOtherStuff();
}
else if(number == 2) {
    doTwiceAsMuchStuff();
}
else {
    iThinkYouGetTheIdea();
}
```

Good

```
switch ( number )  {
    case 0:
        doStuff();
        break;
    case 1:
        doOtherStuff();
        break;
    case 2:
        doTwiceAsMuchStuff();
        break;
    default:
        iThinkYouGetTheIdea();
}
```

Tips

- Don’t use ‘magic numbers’
  - Create some ‘constant’ variables instead

```
var STATE_SPLASH = 0;
var STATE_GAME = 1;
var STATE_GAMEOVER = 2;

var gameState = STATE_SPLASH;

switch (gameState) {
   case STATE_SPLASH:         // process the splash screen state
      break;
   case STATE_GAME:           // process the game state
      break;
   case STATE_GAMEOVER:   // process the game-over state
      break;
}
```

Functions

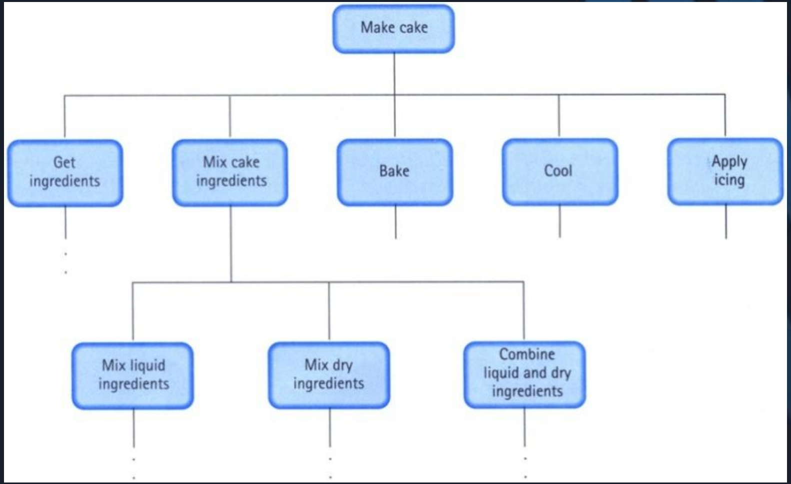
- A block of code designed to perform a task
- Can take input
- Can return output
- Executed when something ‘calls’ it
- We’ve already used some functions

Why do we use functions?

Computer programs are generally very large and consist of complicated problems

The best way to solve the problems is to break them down into smaller, simpler problems.

Also useful for pieces of code that are used multiple times in a program



Declaring a Function

Function declarations have three main parts:

```
function functionName(parameter1, parameter2)
{
	//function body
	return parameter1 * parameter2;
}
```

- **Function name** – A unique identifier for the function (in the same way that variables have unique names)
- **Parameters** – This is the data being passed into our function
- **Function body** – The code to be executed

Functions

Here are some functions you have already dealt with.

These functions have been set up in the assignment template for you to use

```
function getDeltaTime()
{
    ...
}

function run()
{


}
```

Calling a Function

To use a function, it must be called!

To call a function, use the function name, followed by arguments enclosed in parentheses.

The function name should be descriptive (verbs) they should help describe what they are going to do.

```
function updatePlayer(deltaTime)
{
    player.positionX += playerSpeed * deltaTime;
}

function drawPlayer()
{
     context.drawImage(player,
           player.positionX, player.positionY);
}

function run()
{
    // get the delta time
    var deltaTime = getDeltaTime();

    updatePlayer(deltaTime);
    drawPlayer();
}
```

Arguments and Parameters

- When we declare a function, we provide a parameter list.
- This is a list of variables that the function needs to get when the function is called.

```
function findLargest(num1, num2)
{
      if( num1 > num2 )
            return num1;
      else
            return num2;
}
```

- You don't use the var keyword in the parameter list

Arguments and Parameters

- When we call the function, we pass it some arguments.
- The number of arguments needs to match the number of parameters.

```
var score1 = 12;
var score2 = 29;

var largest = findLargest(score1, score2);
var largest2 = findLargest(10, 5);
```

- We can pass variables (ie, score1 and score2).
- We can pass constants (ie, the 10 and the 5).

Variable Scope

- Any variable declared in a function only exists within that function.

- This code will not run since y only exists within the function doStuff()

```
function doStuff()
{
        var x = 4;
        var y = x;
}


doStuff();
document.writeln( y );
```

Variable Scope

- This is also true for variables declared inside the function parameters

- This code will also not compile. You cannot access *value* outside of the *printValue* function

```
function printValue( value )
{
    document.writeln( value );
}


function doStuff()
{
    var x = 1;
    printValue(x);
    document.writeln( value );
}

doStuff();
```

Returning values

- Values can be returned from a function
- Use the return keyword to return a value.
- You don't have to return anything if you don't need to.

```
function findLargest(num1, num2)
{
      if( num1 > num2 )
            return num1;
      else
            return num2;
}

function run( )
{
      var biggerNum = findLargest(20, 10);
}
```

findLargest has a return value that is a number.

More Tips

- Use functions within your switch statements
- Will make your code more readable
- Prevents your switch statement becoming too long

Adding Game States

- Create some state variables
  - Don't use 'magic numbers' in your code

```
// define some constant values for the game states
var STATE_SPLASH = 0;
var STATE_GAME = 1;
var STATE_GAMEOVER = 2;

var gameState = STATE_SPLASH;
```

Adding Game States

- Create the functions for each state

```
function gameStateSplash(deltaTime)
{
}

function gameStateGame(deltaTime)
{
}

function gameStateGameOver(deltaTime)
{
}
```

Adding Game States

- Add the switch statement to the run() function

```
function run()
{
     var deltaTime = getDeltaTime();

     switch( gameState )
     {
          case STATE_SPLASH:
               gameStateSplash(deltaTime);
          break;
          case STATE_GAME:
               gameStateGame(deltaTime);
          break;
          case STATE_GAMEOVER:
               gameStateGameOver(deltaTime);
          break;
      }
}
```

Adding Game States

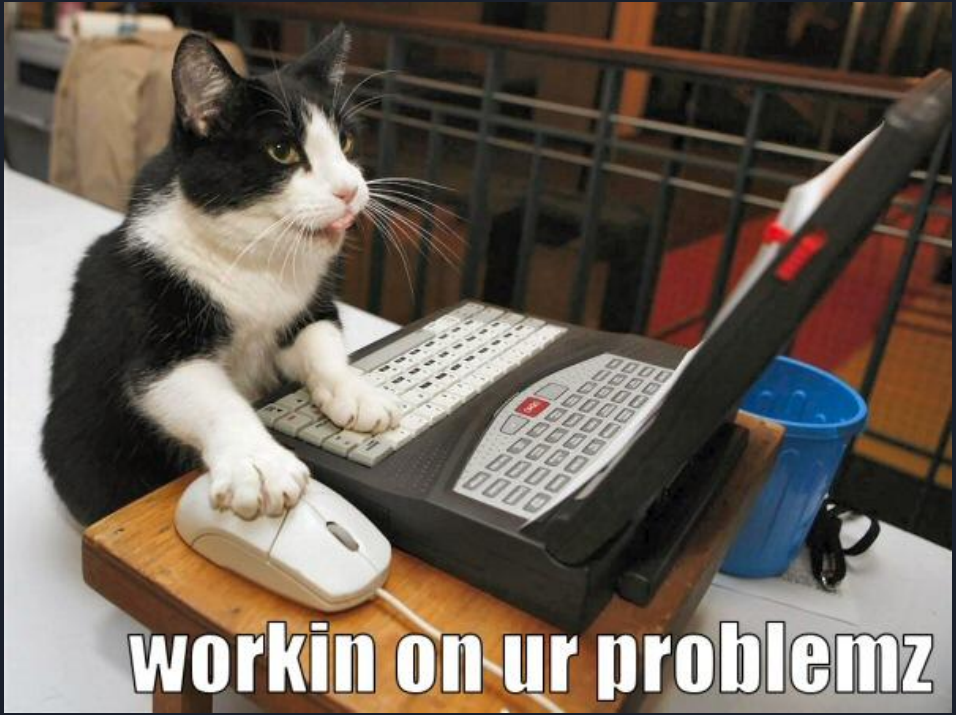
- Finally, fill in each game state function

```
function gameStateGame(deltaTime)
{
     updateShip(deltaTime);
     updateAsteroids(deltaTime);

     drawShip();
     drawAsteroids();

     if (asteroids.length == 0)
     {
            // no more asteroids, end the 'game'
            gameState = STATE_GAMEOVER;
     }
}
```

Summary

- We can use switch statements in our game to control which game state executes
- Functions are blocks of code that perform a single task, can take parameters, and can return a value
- Create functions to break your game into smaller, more manageable pieces

Questions?



aie

SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

References

- JavaScript Switch Statement. 2016. *JavaScript Switch Statement*. [ONLINE] Available at: http://www.w3schools.com/js/js_switch.asp. [Accessed 01 March 2016].
- JavaScript Functions. 2016. *JavaScript Functions*. [ONLINE] Available at: http://www.w3schools.com/js/js_functions.asp. [Accessed 01 March 2016].