Lesson Plan >

Lesson ID: Game Development with HTML5
Faculty Member: Sean Morrow
Subject: Workshop

Content:
HTML5 Game Engine
Spritesheets
Introduction to Adobe Flash
Designing sprites in Flash
Challenge : Design
Animating sprites in Flash
Challenge : Animate
Challenge : Design All Assets
Generating Spritesheets with ZOE
Running the game
Adjusting Game Settings
Selecting Sound effects
Challenge : Code and Polish

Setup:
> gameEngineV4.0 folder deployed on desktop
> Adobe Flash CC opened to gameAssets.fla with playback head on "playerMoving" blank keyframe
> ZOE Installed (Adobe AIR App) and running
> run miniWeb server (will require permissions)
    > using the IP URL provided by miniWeb hit game.htm in chrome
    > save a bookmark to this page on the desktop where the student can easily find it
    > shut down chrome to avoid distraction during workshop but keep miniWeb running

Announcements:
None Required.

Instructional Procedures:
GAME ENGINE
> is a program that takes a collection of assets (sounds, graphics, etc) and brings them all to life in a game
> does all the work for you - in terms of coding
> our game engine is developed in HTML5
    > means it is a combination of HTML / JavaScript
    > it also uses an open source library called CreateJS

> what is the game we will be making?
    > demo on whiteboard with sketch

> main player character goes left and right near bottom of screen
> main player can shoot projectiles (bullets?) but only three can be on the screen at one time (at game start)
> enemies come down from the top
    > move at random angles and at random speeds
> main player must eliminate all enemies before they go off the bottom of the screen
> every enemy that escapes takes a hit point off the main player
> every enemy that touches main player takes a hit point
> main player has five hit points to start (adjustable)
> every enemies eliminated increases the level which means:
    > enemies appear more often with every level
    > the main player regains one hit point
    > the main player has one more projectile that can be on the screen at one time
> if all hit points are taken the game is over

> sound complex?
> the game engine does most of the work
> your main job will be to design:
    > the main player animations (alive, idle, dead)
    > the enemy animations (alive, dead)
    > the projectile
    > the start, in-game, and game over screens
> you will get the chance to play with some code to adjust game settings and add your own sound effects

> you will take away a real appreciation for the amount of work that goes into even a simple game like this one

SPRITESHEETS
> games are made up of graphics that are animated (flying spaceship, zombie, etc)
> these graphics are called sprites
> sprites are collected in giant images of all the frames
> these giant images are called spritesheets
> they quite often contain several sprites
> demo to students with Plane.png
> the game engine takes this giant image, cuts it up and stores individual frames of the animations in memory for use in game
> in order to design our main player, enemy, etc. (sprites in other words!) we need to build a spritesheet of them all

> several ways to build a spritesheet - we will use Adobe Flash
> we will design our sprites in Flash, publish it into a SWF file, and then use a tool called ZOE to convert the SWF into a spritesheet

INTRODUCTION TO ADOBE FLASH
> the first step is how to design (draw) in flash
> how many people have used flash?
> we will use Flash CC but you can use older versions of Flash as well
> flash is a vector based animation tool
    > you can build full games with Flash alone, but not well supported
    > vector based means graphics are stored as math (y = mx + b)
    > opposed to bitmap based graphics (MS Paint, PhotoShop, etc)

> point out different components of Flash:
Tools Panel
> made up of drawing tools
> very similar to MS Paint

Stage
> where you draw your sprites and animate

Property Panel
> is an adaptive panel - changes with whatever is selected
> demo clicking on the stage and then clicking on a timeline keyframe

Timeline Panel
> where you adjust what your sprite looks like over time (animate!)

> start by exploring tools in the tools panel:
    > note to instructor : demo all the tools by drawing a blobby character; something simple in the blank keyframe under the “playerMoving” frame label

    Line Tool:
    > draws straight lines as you hold down the mouse button and drag
        > demo to students
        > show how to change the thickness of the line, color in property panel from color palette
        > only uses the stroke color - talk about stroke and fill color

    Rectangle Tool:
    > draws rectangles/squares
    > to draw a rectangle with equal length sides (square) hold SHIFT key
        > demo to students
        > show how you can change the thickness of the outer line
        > show how to change the color of the stroke and fill with the property panel

    Oval Tool:

> draws circles, ovals
> to draw a perfect circle hold down SHIFT key
    > demo to students
    > show how you can change the thickness of the outer line
    > show how to change the color of the stroke and fill with the property panel

Pencil Tool:
> pencil tool draws any shape that you want; good for small details
> three options:
    > demo to students
    > straighten: converts the lines you draw into a connected straight line segment
    > smooth: converts lines into smooth curves
    > ink: will keep line as close as possible to what you drew, with a few touch ups; jagged lines

Paint Bucket:
> a fill is a color that fills the inside of closed objects (shapes)
> oval, or rectangle tool make these objects
> flash will automatically fill the inside of these objects with whatever currently selected fill color, if want none, choose the no fill color (red stroke color)
    > demo to students
    > draw a filled rectangle
    > change the fill color with the paint bucket
    > change the fill color by selection the whole shape and changing the fill color in the property panel

Selection Tool:
> demo how to drag a selection box around an object to select both fill and strokes
> demo delete button to erase
> demo how to move a shape once drawn
> we are manipulating the math of the graphics

CHALLENGE : DESIGN
> have students design the main player (the character that you will be controlling in your game) using the techniques covered on the “playerMoving” frame of gameAssets.fla
> rules to remember:
    > don't make main player too big (2.5cm x 2.5cm)
    > design the main player facing left (engine will handle facing right)
    > keep the top left corner aligned with the little circle (registration point)

ANIMATING SPRITES IN FLASH

> animation is what Flash does best
    > Flash is used for a lot of TV production cartoons
> animating is the process of making an image change over time
> in terms of our main player:
    > the movement of your main player will be handled by the game engine
    > what we need to animate is what it does *while* it moves

> have class look at current sprite design
> the sprite in flash is made up of smaller separate pieces; hard to animate with so many pieces
> your sprite is known as a Shape in "flash speak"
> we need to group them together so they aren't all these little pieces
> the easiest way with the most flexibility is to convert the shape to a graphic symbol
> go to modify / convert to symbol / graphic symbol [give name]
> point out blue outline around entire sprite and can be moved as one entity

> animation is controlled by the timeline panel
> provides an ability to adjust the frame of time so the stage can update accordingly
> point out layers (explain arrangement)
> point out playback head
> point out frames
    > the game engine requires the animation to run at 30 frames per second
    > means the playback head will move over 30 frames per second
> to animate our main player sprite we need to add keyframes

> two types of frames:
    KEYFRAMES
    > always has a dot in it on the timeline
        > if it's an empty dot there is no shape / group / etc on the timeline/stage on this layer; filled there is something
    > first frame of any object on the timeline is a keyframe
    > they represent crucial moments in time of the animation
        > flash can take keyframes and figure out all the inbetween frames on its own
        > process called Tweening (more in a second)

    ORDINARY FRAMES
    > only flash can add items to these frames via tweens (motion or shape)
    > the developer does all the work in the keyframes and flash does its work in the ordinary frames

> demo to students:
    > add a keyframe 10 frames to the right of the first keyframe
    > click between two keyframes - explain these will need to be different

> select second keyframe
> change the rotation of the sprite using free transform tool to the right
> right click and add classic tween
> tweening is the process of taking two keyframes and calculating inbetween frames
> disney classic animation teams were made up of keyframe animators (veterans) and tween animators (noobs!)
> press ENTER to demo running animation
> you can tween all sorts of changes:
> position, size, skew, color, alpha, etc.

> add a third keyframe and rotate the game character back to left past starting rotation
> add second classic tween

> if time permits frame by frame animation could be demonstrated to show students how to do more complicated animation that cannot be tweened

CHALLENGE : ANIMATE
> design the animation that is to occur while your main player moves to left
> game engine will mirror reverse it to handle when it moves to the right

> instructor needs to assure the frame label in the timeline to match up with the length of the animation; use F5 and SHIFT+F5 to add and remove frames

CHALLENGE : DESIGN ALL ASSETS
> we have loads of work to do now!
> you need to design and animate:
> the main player when it is killed
> we are now adding more animations; each one must be labeled
> demo to students the animation names layer at top of timeline
> demo how to add blank keyframe and label in property panel
> label "playerDead"
> the main player when it is idle (not moving) [OPTIONAL]
> label "playerIdle"
> the enemy when it moves down the screen
> don't make main player too big (2.5cm x 2.5cm)
> design the main player facing left (engine will handle facing right)
> keep the top left corner aligned with the little circle (registration point)
> label "enemyMoving"
> the enemy when it is eliminated
> label "enemyDead"
> the projectile the main player shoots
> will be travelling up so it should point up if it has a front and back to the design

> keep the top left corner aligned with the little circle (registration point)
> label "projectile"

> the start screen (600 x 600)
> will display when the game first loads
> you click anywhere to play the game
> label "screenStart"

> the in-game screen (600 x 600)
> the background on the screen behind all the action
> label "screenInGame"

> the game over screen (600 x 600)
> will display when the game is over
> label "screenGameOver"

GENERATING SPRITESHEETS WITH ZOE
> to publish your sprite animations you press CTRL+ALT
> point out all sprite animations are running in order
> now to convert to a spriteSheet
> demo to students:
> open ZOE and load gameAssets.swf
> will see animation displayed
> we need to set some settings:
> click on settings tab and check off padding and variable frame
> make sure reuse checkbox is unchecked
> click on animations tab and make sure that all animations are working
> the names of the animations correspond to the frame labels added to timeline in Flash

> click on export tab and select the smallest size of spritesheet to produce
> it is more efficient to be in powers of 2 hence the size options
> point out data type set to JSON is the type of data it will produce to correspond with the spritesheet
> JSON is a structure to hold data
> contains information on how your spritesheet needs to be spliced up, the animation sequences, etc.

> click export button in top right corner
> preview gameAssets.png to students

RUNNING THE GAME ENGINE
> on your desktop you will find a bookmark called "Start Game" - double click it to start up the game engine

> most students will want to tweak, regenerate the spritesheet, and test for several minutes

ADJUSTING GAME SETTINGS
> there are a number of game play settings you can change to alter the gameplay of your game
> this involves some actual programming!
> to change them you need to view the script of your game
> demo to students with GameSettings.js:
    > open up GameSettings.js with any text editor
    > point out different settings and go over details
    > demo altering number of lives
    > re run game to demo changes made
    > be careful altering these as game settings a big part of proper game balance - too easy or too hard
> notice the sound effects at the bottom

ADDING SOUND EFFECTS
> the final step is adding some sound effects
> there are a number of built in sound effects in the engine to choose from in the sound folder
> each sound effect is stored in both MP3 and OGG (the two main formats required for HTML5 games)
    > different browsers require one of these formats for playback
> you can sample any sound by double clicking on any mp3 (requires some sort of media player)
> to adjust a sound you need to change some game settings
> demo to students with GameSettings.js:
    > change following line:
    "SoundFireBullet":"gunFire"
    > re run game to demo changed sound effect
    > point out name of sound effect is same as filename minus extension

> it is possible that the students can download their own sound effects but they must include it in the sounds folder as both MP3 and OGG
> Audacity could be used for converting the two
> the sound effect will need to be added to the Manifest.js – for example:

```
{
    src:"sounds/zip.ogg",
    id:"zip",
    data:4
}
```

> it is possible that all the games developed could be deployed on a server so they can be played from home, etc…

Materials and Equipment
HTML5 Game Engine folder
Plane.png
lessonResources/gameAssets.fla (a completed gameAssets for reference)