Examples for GameDev Library V4

This document explains the example that are in this folder. The examples are some of the more common features of the Gamedev Library. I’ve organized it to explain the “Object Data Types” and functions available to you and how the examples in this folder work.

These examples require certain images and sound files exist in the images and sound folders.

In each example I’ve surrounded all the unique code for the example with comments that look like

//\*\*\*\*!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

## Stepwise development

Simply getting all of your game elements ( background, Sprites, SpriteSheets, GameText) drawing on the screen is the first step you should take in creating your game. Don’t stress over where they draw, if the Spritesheet has movement, you can check for collision, etc.

A next step on this would be to create any arrays of Objects you think you’ll want – such as items to collect, bulllets/missles; terrain; enemies..

Next organize these objects on the page where you ultimately want them to be.

Then capture keyboard/mouse events to create basic movement of your character around the screen,

Then add basic collision detection. To be fair – collision detection can be the hardest part of your game to code – so that is why you should leave this for last.

# GameText

GameText objects allow you to draw words in specific locations, sizes, colors, and fonts on the canvas.

This is most often used to display scores, lives remaining, etc.

The example for using GameText can be found in GameText\_Template.html

In the setup ( initialization) portion of the game we do the following:

var score = new GameText();

score.x = 50; //changes the default x coordinate of the text

score.y = 100; //changes the default y coordinate of the text

score.font = "32px Arial"; //changes the default font and size fo the text

score.fillStyle = "Red"; //changes the default color of the text

score.alpha = 0.5; //changes the transparency of the text

the first line creates the score “object” called score

the remaining lines change some of its properties of score. Note you don’t have the change these if you want to keep the defaults. Also note that you can change these at any time during the game if you like.

In the draw loop in step #3 we draw the score object as such:

score.draw("The Score is:" + game.gameIterator);

Note that we pass as a parameter to the draw method the text we want to print. In this case the text “The Score is:” + the gameIterator ( which is just a counter of every time the screen repaints…

If you want to draw more than one pieces of text you should create multiple GameText objects and do the same for them.

# Sprite

Sprites are simply individual images you want to draw on the screen. Use these for images that have no motion. Note an image may still have motion – even if it doesn’t move around the screen. For example, we’ll see in the SpriteSheet example a campfire that stays in one place, but the fire itself is shown “burning”.

The example for using Sprite can be found in Sprite\_Template.html

## In the setup ( initialization) portion of the game we do the following:

var imgCup = new Image();

imgCup.src = game.imageDir + "cup.png";

var cup = new Sprite(83,100,500,500,200,200,imgCup);

var imgBird = new Image();

imgBird.src = game.imageDir + "bird.png";

var bird = new Sprite(200,120,225,225,100,100,imgBird);

Note that for both images I FIRST create an image variable to hold the image. I tend to name that image the same as the Sprite Object I’ll use it in, but with the prefix img.

Then I load the image into that variable by putting the file name in the .src property. Note that I concatenate the imageDir before the image file name – which indicates the image is in the image directory.

Once I have the image variable created and loaded I can create my Sprite Object. The parameters that go into creating the Sprite are as follows:

1. Starting x position of the Sprite
2. Starting y position of the Sprite
3. Width of the image in pixels\*
4. Height of the image in pixels\*
5. useWidth ( width you want to use when drawing the image)
6. useHeight ( Height you want to use when drawing the image)
7. the image variable you already created.

\*- To get the “actual” height and width of an image right -click on it – click Prpoerties, then look in the details tab

## To draw the image you simply call its draw method – like:

cup.draw();

bird.draw();

Note the following:

By default the Sprite’s visible property is true. But if you set it to false it will not be drawn.

Also note that both the cup and the bird have portions of their image that are transparent. I purposefully drew the bird on top of the cup to demonstrate this.

# SpriteSheet

SpriteSheets are for extensions of Sprites. The big difference is that their image files are actually a collection of images that you change rotate between to give your object the appearance of motion.

These generally come in two varieties – I’ve included both versions in this example:

* Single “row” SpriteSheets that are meant for images that show movement, but the motion does NOT change “directions” over time.
* Multiple “row” SpriteSheets where the movement “direction” will change . This is most likely a character that can move left or right ( or even up and down and diagonals). The process of creating the SpriteSheet is the same, it is just that with single row versions you won’t ever change the “Direction” .

The example for using Sprite can be found in SpriteSheet\_Template.html

## In the setup ( initialization) portion of the game we do the following:

var imgFire = new Image();

imgFire.src = game.imageDir + "CampFire.png";

var fire = new SpriteSheet(200,200,64,64,128,126,1,4,imgFire);

var imgMario = new Image();

imgMario.src = game.imageDir + "mario.png";

var MARIO\_HEIGHT = 128;

var MARIO\_WIDTH=94;

var mario = new SpriteSheet(0,canvas.height-100,47,64,MARIO\_WIDTH,MARIO\_HEIGHT,3,5,imgMario);

mario.dX = 0;

mario.dY = -2;

mario.currentDir = 0;

Note that for both images I FIRST create an image variable to hold the image. I tend to name that image the same as the Sprite Object I’ll use it in, but with the prefix img.

Then I load the image into that variable by putting the file name in the .src property. Note that I concatenate the imageDir before the image file name – which indicates the image is in the image directory.

Once I have the image variable created and loaded I can create my Sprite Object.

This is similar to the Sprite, but there are a couple of key differences to look out for…

The parameters that go into creating the Sprite are as follows:

1. Starting x position of the Sprite
2. Starting y position of the Sprite
3. Width of the image in pixels\*
4. Height of the image in pixels\*
5. useWidth ( width you want to ue when drawing the image)
6. useHeight ( Height you want to use when drawing the image)
7. maxDir – the number of Rows – but this is an index – so it is number of rows -1
8. maxFrame – number of “frames” or columns in each row – again, an index so it is columns -1
9. the image variable you already created.

\*- To get the “actual” height and width of an image right -click on it – click Properties, then look in the details tab. But wait – there is more… This is the height and width of the ENTIRE Spritesheet ( all rows and columns). We want to pass the height and width of an individual frame – so we need to do some math.

* Take the height and divide it by the number of rows – this is the height to pass.
* Take the width and divide it by the number of frames – this is your width

The two additional parameters are:

* The number of “directions ( ie rows) MINUS 1.
* The number of “frames” in each row MINUS 1

We subtract one from each because this is the Maximum INDEX, and as we know, indexes start at ZERO

Also note that the SpriteSheet object will start at the first Row, so if you want it to start at something different you need to change the <objectname>.curentDir to whatever row you want. This is true anytime in the game you want to change “directions” as well.

## To draw the image you simply call its draw method – like:

fire.draw();

mario.draw();

HOWEVER, notice I’ve added some more code to both control how fast the frames change AND control Mario moving around the screen. Explanation of that below…

if (game.gameIterator%5==0){ // I want the fire frame to change every 5th page repaint

fire.advanceFrame();

}

fire.draw();

if (game.gameIterator%3==0){ //I want Mario frame to change every 5th repaint

mario.advanceFrame();

}

if (mario.y<0){ //mario was moving up at left - change to move right

mario.y = 0;

mario.dX = 3;

mario.dY = 0;

mario.currentDir = 1;

}

else if (mario.x<0){ //mario was moving left at bottom - change to move up

mario.x = 0;

mario.dX = 0;

mario.dY = -3;

mario.currentDir = 0;

}

else if (mario.x>canvas.width-MARIO\_WIDTH){ //mario was moving right at top - change to move down

mario.x = canvas.width-MARIO\_WIDTH;

mario.dX = 0;

mario.dY = 3;

mario.currentDir = 2;

}

else if (mario.y>canvas.height-MARIO\_HEIGHT){ //mario was moving down at right - change to move left

mario.y = canvas.height-MARIO\_HEIGHT;

mario.dX = -3;

mario.dY = 0;

mario.currentDir = 3;

}

mario.x+=mario.dX;

mario.y += mario.dY;

mario.draw();

The advanceFrame() method of both SpriteSheets will move to the next frame in the object and will circle back to the first if it is already on the last frame. Notice I put these within IF statements that slow both down a bit.

The rest of the code moves Mario around the screen to demonstrate how he looks moving.

Note the following:

By default the Sprite’s visible property is true. But if you set it to false it will not be drawn.

Also note that both the cup and the bird have portions of their image that are transparent. I purposefully drew the bird on top of the cup to demonstrate this.

ScrollingBackGround

The ScrollingBackGround Data Type is much like a regular Sprite, but it has the ability to scroll in the x and y directions and will wrap around itself so that as part of it scrolls off the screen that same part of the image appears on the opposite side of the screen. Think f it kind of like a cylinder. Makes more sense when you see it in action.

The example for using ScrollingBackGround can be found in ScrollingBackGround \_Template.html

You can, of course , not have the background scroll at all – it is just an option.

Another option related to this is that you can have Sprites and Spritesheet objects scroll with the background. That is part of this demonstration as well.

The following is the code for creating both the background and the bird object…

var imgClouds = new Image();

imgClouds.src = game.imageDir + "cloudsBG.png";

var bg = new ScrollingBackGround(0,0,320,256,canvas.width,canvas.height,imgClouds);

var imgBird = new Image();

imgBird.src = game.imageDir + "bird.png";

var bird = new Sprite(200,120,225,225,100,100,imgBird);

bird.moveWithBackground = true; //Note I set this to true so the bird will scroll with the background

While you can set this at any point. I’ve also set the scroll rate for the background just after I created the game object..

var game = new GameMaster();

//Variables that determine how fast (and if) the background moves. If the background

//does NOT move set both to 0 - this is the default;

game.bgDX = 1;

game.bgDY = 0;

The parameters that go into creating the ScrollingBackGround are as follows:

1. Starting x position of the Sprite (always 0)
2. Starting y position of the Sprite (always 0)
3. Width of the image in pixels
4. Height of the image in pixels
5. useWidth ( set this equal to the canvas.width)
6. useHeight ( set this equal to the canvas.height)
7. the image variable you already created.

Finally… worth pointing out that the background is the FIRST thing you draw once the screen is cleared - otherwise it will simply be drawn over top of your other objects…

bg.draw();

bird.draw();

createAudio

createAudio is a function that just takes some of the steps out of creating a sound. To use it you simply pass the audiofile location/name to the function. By default, your audio files should be in the sounds

folder just off of your game.

The code for this example JUST plays a sound when you press the right key. There are not objects drawn to the screen.

if (myKeys.rightArrow) {

if (game.gameIterator%19==0){

createAudio(game.soundDir +'pacman\_chomp.wav');

}

}

Notice I also put createAudio in an IF statement to ensure it doesn’t play too often.

The audio file pacman\_chomp.wav must be in the sounds directory off of where your game is for this to work..

Collision Detection

createAudio is a function that just takes some of the steps out of creating a sound. To use it you simply pass the audiofile location/name to the function. By default, your audio files should be in the sounds

folder just off of your game

Every Sprite and Spritesheet object has some built-in methods that allow you to detect if it is ( or will be) in collision with another Sprite or Spritesheet. These are called

. checkBottomCollision(obj2)

.checkTopCollision(obj2)

.checkRightCollision(obj2)

.checkLeftCollision(obj2)

In our example I’m checking for a collision BEFORE I let Mario move into the fire – except when moving down…

if (myKeys.rightArrow) {

if (mario.checkRightCollision(fire)){

txtRight.draw("Right Collision");

}else {

mario.currentDir = 1;

mario.x++;

}

}

if (myKeys.leftArrow) {

if (mario.checkLeftCollision(fire)){

txtLeft.draw("Left Collision");

} else{

mario.currentDir = 3;

mario.x--;

}

}

if (myKeys.upArrow) {

if (mario.checkTopCollision(fire)){

txtTop.draw("Top Collision");

}else{

mario.currentDir = 0;

mario.y--;

}

}

if (myKeys.downArrow) {

//if (mario.checkBottomCollision(fire)){

// txtBottom.draw("Bottom Collision");

//}

mario.currentDir = 2;

mario.y++;

}

KeyPresses

Keypresses capture MOST of the types of keyboard and mouse inputs you may want to utilize in your game. In shirt, they keep track of every key on the keyboards and tell you if it is currently pressed or not. Likewise, it keeps track of the mouse position and whether any of the mouse buttons have been clicked or not.

Capturing keys was already demonstrated in CollisionDetections, so I won’t bore you with another example…