Chapman University
CPSC 340 - Spring 2011

Class #4
September 22, 2011
William Wood Harter
Adjunct Professor

Summary

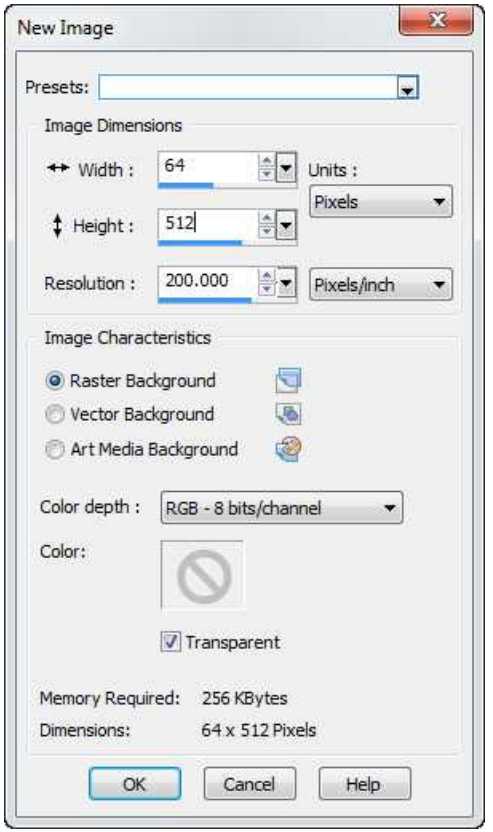
- Var names (case)
- Comments and Copyrights
- Sprite sheets
- Zelda Tile Systems

Variable and Method Naming

- In C# it is recommend in naming variables.
  - start with lower case letters and use camel case
    - currentCell
- Method calls start with a capital letter
- Class names start with an upper case letter
- As I said before, I also like to start a variable name with 1 to 3 characters that identify the type.
  - vLocation

Creating A Sprite Sheet

- Make your sprite sheets align vertically, not horizontally. Explanation coming.
- Here is an 64 pixel sprite with 8 (64*8=512) images.

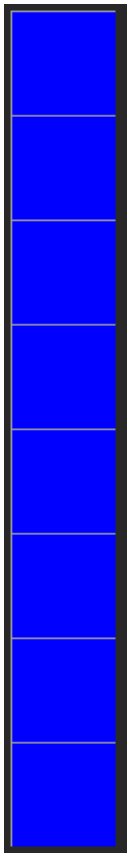


Why Align Vertically

- Memory is the reason.
- Remember when we talked about cache.
- All cells in the sprite must be uniform size.
- Try to keep it a power of two it may help the cache.

Image Guides

- Use image guides to help see where each image is.
- I've setup my 1x8 sprite sheet to have horizontal guides every 64 pixels.



An Explosion SpriteSheet

- Drawn in less than 2 minutes Not pretty, but it works as programmer art
- 8 cells of 64 pixels each
- Transparency included
- Saved as a png

How To Draw

- The trick is to only draw part of the sprite image to the screen. The cell you want to draw.

```
public void Draw(SpriteBatch sb)
{
    if (CurrentCell >= 0)
    {
        sb.Draw(image, location, sourceRect, Color.White);
    }
}
```

- Wait, I thought sb.Draw only had three parameters?

```
spriteBatch.Draw(texOver, location, Color.White);
```

- There are a numerous ways to call Draw
- http://msdn.microsoft.com/en-us/library/microsoft.xna.framework.graphics.spritebatch.draw.aspx

How To Draw

- This call to Draw expects a location vector to draw at on the screen and a sourceRect of where to take the image from in the source.

```
public void Draw(SpriteBatch sb)
{
    if (CurrentCell >= 0)
    {
        sb.Draw(image, location, sourceRect, Color.White);
    }
}
```

- Notice CurrentCell, it is the vertical cell number. The rectangle is built from that whenever CurrentCell is changed.

```
// set the current cell and update the source rectangle for the sprite sheet
private int currentCell;
public int CurrentCell
{
    set
    {
        currentCell = value;
        sourceRect.X = 0;
        sourceRect.Width = cellWidth;
        sourceRect.Y = currentCell * cellHeight;
        sourceRect.Height = cellHeight;
    }
    get { return currentCell; }
}
```

Getters and Setters discussed on the next slide.

Getters And Setters

- An object oriented trick that helps in code reuse is to use Getters And Setters. It's a good practice.
- Keep your variables private and create getter and setter methods to set the variables.
- This also allows you to change other values when the value changes.

```
// set the current cell and update the source rectangle for the sprite sheet
private int currentCell;
public int CurrentCell
{
    set
    {
        currentCell = value;
        sourceRect.X = 0;
        sourceRect.Width = cellWidth;
        sourceRect.Y = currentCell * cellHeight;
        sourceRect.Height = cellHeight;
    }
    get { return currentCell; }
}
```

Automating Cell Change

- You can automate the change of the cell.
- Add an update method and call it from your update.
- Keep a tick count since the last change.
- You might want to configure the number of ticks between changes instead of hard coding.
- Every time through update, decrement the tick count. If it is less than 0, go to the next cell.
- When you reach the last cell, repeat.

Ticks or Time

- Ticks is my preferred method to change cells.
- A tick is just a call to Update.
- This might be faster or slower on different computers or platforms. You might need to measure and adjust.
- If you use Time, you might skip showing cells during machine “blips”. Both ways have positives and negatives.
- At the end of the day it is your preference.

Always Repeat?

- It might not be best to always repeat.
- You might want to add a repeat count
- repeat count -1 is common for repeat forever.
- If repeat is a number, need a way to reset the sprite so it plays again.
  - Reset simply means resetting the repeat count.

SpriteSheet.Update

```
public void Update(GameTime gameTime)
{
    // see if ticksTillChange has fallen below zero
    if (ticksTillChange < 0)
    {// move to the next cell if we are below the number of cells
        if ((CurrentCell < numCellsHigh) && (CurrentCell>=0)) {
            CurrentCell = CurrentCell + 1;
            ticksTillChange = ticksBtwnImages;
        }else
        { // we are beyond the last cell, check repeat count -1 = reoeat forever
            if (repeatCount == -1){
                CurrentCell = 0;
                ticksTillChange = ticksBtwnImages;
            }else
            {
                if (currentRepeat > 0) {
                    // otherwise only go the specified number of repeats
                    currentRepeat--;
                    CurrentCell = 0;
                    ticksTillChange = ticksBtwnImages;
                }else
                {// repeat count is done, set the current cell off the image so it doesn't draw anymore
                    currentCell = -1;
                }

            }
        }
    }else
    {// just decrement the tick counter
        ticksTillChange--;
    }
```

Sample SpriteSheet App

- I wrote a sample SpriteSheet application you can download and try out.
- It isn't fancy, just plays through one very ugly sprite sheet that I made in a few minutes.
- Press the A key to play through the sprite.

Noticing bugs quickly

- If you find that I notice bugs quickly it's only because I probably spent three days looking at the same or similar bug at some point.
- Programming is not about intelligence
- It's all about the amount of experience
- It is said that it takes 10,000 hours to gain expertise in any topic.

Zelda Tile Systems

- GameBoy Advance had a built in hardware tile system.
- As I remember, you had 64 different 8x8 tiles you could use.
- Your screen map x+1 tiles wide and y+1 tiles high.
- To scroll around you set the offset, then had to change the edges if you shifted into the next tile area.
- The tile are wrapped around

How to mimic that tile system in XNA

- Create an array of images.
- Could be merged into a single SpriteBatch
- Your map is simply a list of integers that index into the SpriteSheet.
- The array list size is the number of tiles that fit in the x direction +1 and y direction +1.
- You code goes through the array of tiles and draws the image index for each screen location.
- Include an offset if you want the screen to scroll

Comments and Copyrights

- You should put a code header in the top of all your files.
- It should include your name and a copyright notice like this
  - (c) 2011 William Wood Harter – All Rights Reserved Worldwide
- It should also include a description of the file.
- Copyrights are good for life+50 years.

Movement with keyboards

- You know how to get keyboard input
- Can you move an image left and right based on the A and D keys?

Handling Bullets

- They are just like moving the ship left and right, but they move in a specific direction at regular intervals (ticks)
- They are created by a keyboard event.
- There may be an array of bullets

Handling Collision

- You can simply check to see if a point is in a rectangle or within a specific distance.
- Use round or square things to make your life easier.
- The rectangle collision is the same as the button click code.
- The circle collision is simply distance between two points
    - sqrt($(x2-x1)^2 + (y2-y1)^2$)

Assignment #6

- Create image that moves left and right using the A and D keys. Place the image at the bottom of the screen.
- When the space bar is pressed, create a bullet that travels up towards the top of the screen at a specific interval.
- Create a target that moves from left to right at a regular speed at the top of the screen. If the bullet hits the target, give the player a point. When the target moves off the screen start it's position over. When the bullet moves off the screen allow the user to shoot again.
- Change the count up timer to be a count down timer. Give the player a specific amount of time to hit the target as many times as he can. Once time is up, return to the menu. Display the score and time somewhere on the screen.
- This is mostly just using code we've already written and a bit of code from the lecture. Don't make it harder than it is.

Open Time

- Feel free to ask hard questions
- Feel free to work on your home work and ask questions while I'm still here
- Feel free to use this time for your own benefit