

XNA Basics

Vectors

This tutorial is concerned with the math that makes this code work more than displaying what is happening. To get a better handle on what math is used in this code and other math skills that can be applied go to the web site and download the Vector Math pdf file. Look for that file in the Vectors section of the XNA Basics series.

To begin a new project needs to be created.

File ⇒ New ⇒ Project ⇒ XNA Windows Game Studio 3.1 ⇒ Windows Game (3.1)

Give the project a name and decide where you want it to reside on your computer.

A good practice after creating a new project is to change Game1.cs in the Solution Explorer to something more descriptive of the project.

```
public class vectors : Microsoft.Xna.Framework.Game
{
    GraphicsDeviceManager graphics;
    SpriteBatch spriteBatch;
    Vector2 enemyPosition;
    Vector2 playerPosition;
    Vector2 bulletVelocity;
    Vector2 bulletPosition;

    public vectors()
    {
        graphics = new GraphicsDeviceManager(this);
        Content.RootDirectory = "Content";
    }
}
```

Vector2 objects need to be declared so that we are to track the position of our objects on screen.

```

/// <summary>
/// Allows the game to perform any initialization it needs to before starting to run.
/// This is where it can query for any required services and load any non-graphic
/// related content. Calling base.Initialize will enumerate through any components
/// and initialize them as well.
/// </summary>
protected override void Initialize()
{
    // TODO: Add your initialization logic here
    enemyPosition = new Vector2(2, 4);
    playerPosition = new Vector2(10, 80);

    bulletPosition = enemyPosition;
    bulletVelocity = new Vector2(playerPosition.X - enemyPosition.X, playerPosition.Y -
enemyPosition.Y);
    bulletVelocity.Normalize();
    base.Initialize();
}

```

Now we need to create the the objects that we declared earlier. Give the enemy ship a name and then create a new object out of it. To insure that is will spawn in the top left corner of the screen use small values like the ones used in the examples.

Next repeat for the player ship and use a large y value so that it will spawn in the bottom of the screen.

Set the bullet's position equal to the enemy's position so that the origin of the bullet will be the enemy ship.

Now we need to do a vector operation so that the bullet will travel in a straight line toward the player ship when it is fired. The only problem with the way the bullet travels toward the player is that it move instantly. We don't want that to happen so we have to add one more line of code.

The last line of added code that says:

```
bulletVelocity.Normalize();
```

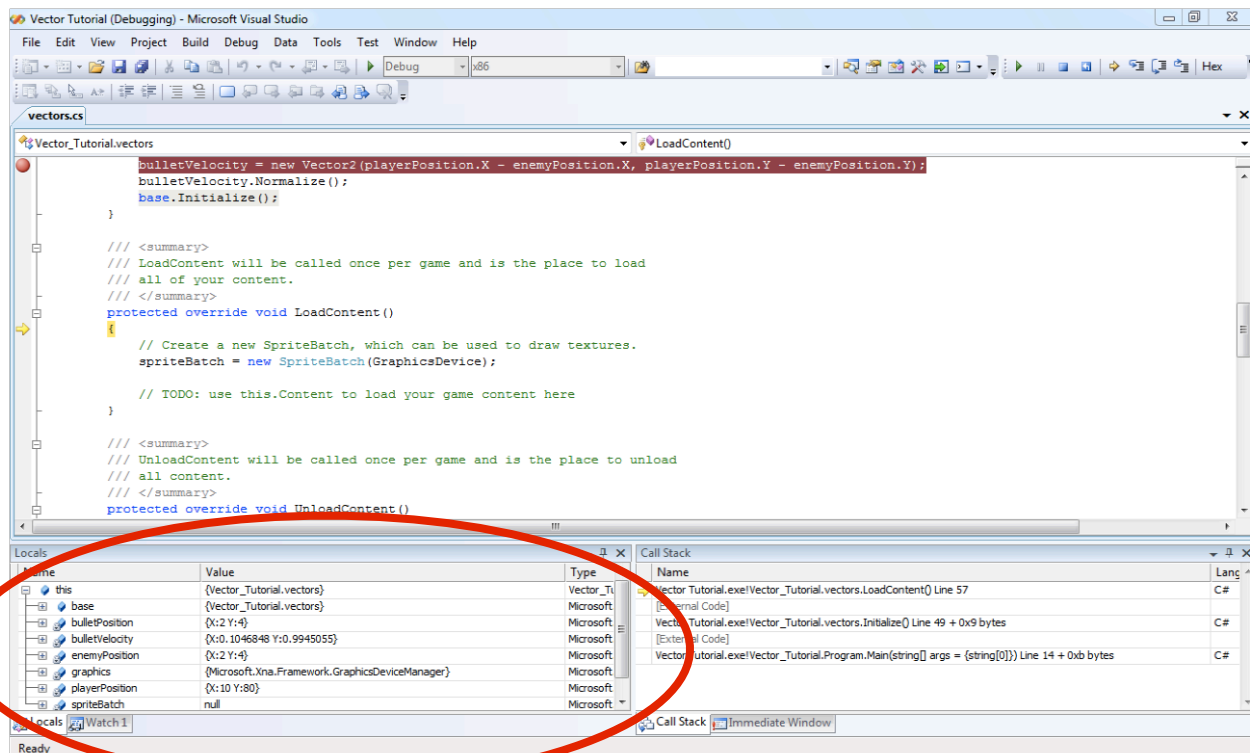
Is the code that makes the bullet travel in one smooth continuous line.

One thing that we have not done in this tutorial is load images for the player and enemy objects, so if you run the program you won't see what is happening. Since this tutorial is only interested in the math behind what is happening not seeing anything is okay.

To see the numbers describing what is happening set a break point at the line where the vector operation happens.

You can set a break point by left clicking in the gutter right next to line that you want to break at.

To see the data that is created by our program start debugging by pressing f5. Once started step into operations by pressing f11. You should see something like this.



This is the part that displays all of the data that we are interested in right now.