Eyes of the Dragon Tutorials
Part 36
Animated Tiles

I'm writing these tutorials for the MonoGame 3.8 framework using Visual Studio 2019. The tutorials will make more sense if they are read in order. You can find the list of tutorials on the Eyes of the Dragon page of my web blog. I will be making each version of the project available on GitHub here. It will be included on the page that links to the tutorials.

In this tutorial I will be updating the map editor to add in some of the new items that have been added to the game, such as animated tiles. Fire up your solution in Visual Studio. Right click the XLevlEditor project, right click it and select Set As StartUp Project.

The first thing that I'm going to tackle is adding a few new items to the main map editor form. You can either follow the manual instructions to design the form or copy and paste the designer code that follows them.

Double click the FormMain form to bring it up in design view. Now, select the Tileset menu item in the designer. Place your cursor in the Type Here box and enter &Animated Tileset. Select the tab control on the right side of the page. In the properties window for the control open the Tab Pages collection. Click the Add button to add a new tab. In the properties for the new tab set its (Name) property to tabAnimated and its text to Animated Tiles. Using the up and down arrows place this tab as the second tab in the list. Now add another new tab with the (Name) property of tabCollisions and the Text property Collision Layer. Place this tab as the fourth tab in the list.

Now, close the tab collection window. In the properties tab select the Animated Tiles tab. Drag a check box onto the tab, then a numeric up down box and a picture box onto the tab. Position the check box at the top of the tab. Set its (Name) property to chkPaintAnimated and its Text property to Paint animated tile. Place the numeric up down under the check box. Set its (Name) property to sbAnimatedTile and the Minimum, Maximum and Value properties to -1. Set the (Name) property of the picture box to pbAnimatedSet. Click on the Anchor property and select all four options. Set its SizeMode property to StretchImage. Now drag and position the box so that it takes up the rest of the tab.

Select the Collision Layer tab now. Drag a check box onto the tab and then a group box. Onto the group box drag two radio buttons. Change the (Name) property of the check box to chkPaintCollision and the Text property to Paint collisions. Change the Text property of the group box to Collision Types. Change the (Name) property of the first radio button to rbNoCollision, the Text property to No Collision and the Checked property to True. Set those same properties for the second radio button to rbImpassable, Impassable and False.

As promised here is the code for the FormMain.Designer.cs file that you can copy and paste instead of designing the form manually.

```
namespace XLevelEditor
```

```
{
    partial class FormMain
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            this.menuStrip1 = new System.Windows.Forms.MenuStrip();
            this.levelToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.newLevelToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.openLevelToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.saveLevelToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.toolStripMenuItem1 = new System.Windows.Forms.ToolStripSeparator();
            this.exitEditorToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.viewToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.displayGridToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.gridColorToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.blackToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.blueToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.redToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.greenToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.yellowToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.whiteToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.tilesetToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.newTilesetToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.openTilesetToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.saveTilesetToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.mapLayerToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.newLayerToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.openLayerToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.saveLayerToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.charactersToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.chestsToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.keysToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.brushesToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.x1ToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
```

```
            this.x2ToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.x4ToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.x8ToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();
            this.splitContainer1 = new System.Windows.Forms.SplitContainer();
            this.tabProperties = new System.Windows.Forms.TabControl();
            this.tabTilesets = new System.Windows.Forms.TabPage();
            this.tbMapLocation = new System.Windows.Forms.TextBox();
            this.lblCursor = new System.Windows.Forms.Label();
            this.lblTilesets = new System.Windows.Forms.Label();
            this.lbTileset = new System.Windows.Forms.ListBox();
            this.pbTilesetPreview = new System.Windows.Forms.PictureBox();
            this.lblCurrentTileset = new System.Windows.Forms.Label();
            this.nudCurrentTile = new System.Windows.Forms.NumericUpDown();
            this.gbDrawMode = new System.Windows.Forms.GroupBox();
            this.rbErase = new System.Windows.Forms.RadioButton();
            this.rbDraw = new System.Windows.Forms.RadioButton();
            this.lblTile = new System.Windows.Forms.Label();
            this.pbTilePreview = new System.Windows.Forms.PictureBox();
            this.tabLayers = new System.Windows.Forms.TabPage();
            this.clbLayers = new System.Windows.Forms.CheckedListBox();
            this.tabCharacters = new System.Windows.Forms.TabPage();
            this.tabChests = new System.Windows.Forms.TabPage();
            this.tabKeys = new System.Windows.Forms.TabPage();
            this.controlTimer = new System.Windows.Forms.Timer(this.components);
            this.animatedTIlesetToolStripMenuItem = new
System.Windows.Forms.ToolStripMenuItem();
            this.tabAnimated = new System.Windows.Forms.TabPage();
            this.tabCollisions = new System.Windows.Forms.TabPage();
            this.chkPaintAnimated = new System.Windows.Forms.CheckBox();
            this.pbAnimatedSet = new System.Windows.Forms.PictureBox();
            this.chkPaintCollision = new System.Windows.Forms.CheckBox();
            this.groupBox1 = new System.Windows.Forms.GroupBox();
            this.rbNoCollision = new System.Windows.Forms.RadioButton();
            this.rbImpassable = new System.Windows.Forms.RadioButton();
            this.mapDisplay = new XLevelEditor.MapDisplay();
            this.sbAnimatedTile = new System.Windows.Forms.NumericUpDown();
            this.menuStrip1.SuspendLayout();
            ((System.ComponentModel.ISupportInitialize)(this.splitContainer1)).BeginInit();
            this.splitContainer1.Panel1.SuspendLayout();
            this.splitContainer1.Panel2.SuspendLayout();
            this.splitContainer1.SuspendLayout();
            this.tabProperties.SuspendLayout();
            this.tabTilesets.SuspendLayout();
            ((System.ComponentModel.ISupportInitialize)(this.pbTilesetPreview)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this.nudCurrentTile)).BeginInit();
            this.gbDrawMode.SuspendLayout();
            ((System.ComponentModel.ISupportInitialize)(this.pbTilePreview)).BeginInit();
            this.tabLayers.SuspendLayout();
            this.tabAnimated.SuspendLayout();
            this.tabCollisions.SuspendLayout();
            ((System.ComponentModel.ISupportInitialize)(this.pbAnimatedSet)).BeginInit();
            this.groupBox1.SuspendLayout();
            ((System.ComponentModel.ISupportInitialize)(this.sbAnimatedTile)).BeginInit();
            this.SuspendLayout();
            //
            // menuStrip1
            //
            this.menuStrip1.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {
            this.levelToolStripMenuItem,
            this.viewToolStripMenuItem,
```

```
            this.tilesetToolStripMenuItem,
            this.mapLayerToolStripMenuItem,
            this.charactersToolStripMenuItem,
            this.chestsToolStripMenuItem,
            this.keysToolStripMenuItem,
            this.brushesToolStripMenuItem});
            this.menuStrip1.Location = new System.Drawing.Point(0, 0);
            this.menuStrip1.Name = "menuStrip1";
            this.menuStrip1.Padding = new System.Windows.Forms.Padding(4, 2, 0, 2);
            this.menuStrip1.Size = new System.Drawing.Size(1151, 24);
            this.menuStrip1.TabIndex = 0;
            this.menuStrip1.Text = "menuStrip1";
            //
            // levelToolStripMenuItem
            //
            this.levelToolStripMenuItem.DropDownItems.AddRange(new
System.Windows.Forms.ToolStripItem[] {
            this.newLevelToolStripMenuItem,
            this.openLevelToolStripMenuItem,
            this.saveLevelToolStripMenuItem,
            this.toolStripMenuItem1,
            this.exitEditorToolStripMenuItem});
            this.levelToolStripMenuItem.Name = "levelToolStripMenuItem";
            this.levelToolStripMenuItem.Size = new System.Drawing.Size(46, 20);
            this.levelToolStripMenuItem.Text = "&Level";
            //
            // newLevelToolStripMenuItem
            //
            this.newLevelToolStripMenuItem.Name = "newLevelToolStripMenuItem";
            this.newLevelToolStripMenuItem.Size = new System.Drawing.Size(133, 22);
            this.newLevelToolStripMenuItem.Text = "&New Level";
            //
            // openLevelToolStripMenuItem
            //
            this.openLevelToolStripMenuItem.Name = "openLevelToolStripMenuItem";
            this.openLevelToolStripMenuItem.Size = new System.Drawing.Size(133, 22);
            this.openLevelToolStripMenuItem.Text = "&Open Level";
            //
            // saveLevelToolStripMenuItem
            //
            this.saveLevelToolStripMenuItem.Name = "saveLevelToolStripMenuItem";
            this.saveLevelToolStripMenuItem.Size = new System.Drawing.Size(133, 22);
            this.saveLevelToolStripMenuItem.Text = "&Save Level";
            //
            // toolStripMenuItem1
            //
            this.toolStripMenuItem1.Name = "toolStripMenuItem1";
            this.toolStripMenuItem1.Size = new System.Drawing.Size(130, 6);
            //
            // exitEditorToolStripMenuItem
            //
            this.exitEditorToolStripMenuItem.Name = "exitEditorToolStripMenuItem";
            this.exitEditorToolStripMenuItem.Size = new System.Drawing.Size(133, 22);
            this.exitEditorToolStripMenuItem.Text = "E&xit Editor";
            //
            // viewToolStripMenuItem
            //
            this.viewToolStripMenuItem.DropDownItems.AddRange(new
System.Windows.Forms.ToolStripItem[] {
            this.displayGridToolStripMenuItem,
```

```
            this.gridColorToolStripMenuItem});
            this.viewToolStripMenuItem.Name = "viewToolStripMenuItem";
            this.viewToolStripMenuItem.Size = new System.Drawing.Size(44, 20);
            this.viewToolStripMenuItem.Text = "&View";
            //
            // displayGridToolStripMenuItem
            //
            this.displayGridToolStripMenuItem.Checked = true;
            this.displayGridToolStripMenuItem.CheckOnClick = true;
            this.displayGridToolStripMenuItem.CheckState =
System.Windows.Forms.CheckState.Checked;
            this.displayGridToolStripMenuItem.Name = "displayGridToolStripMenuItem";
            this.displayGridToolStripMenuItem.Size = new System.Drawing.Size(137, 22);
            this.displayGridToolStripMenuItem.Text = "&Display Grid";
            //
            // gridColorToolStripMenuItem
            //
            this.gridColorToolStripMenuItem.DropDownItems.AddRange(new
System.Windows.Forms.ToolStripItem[] {
            this.blackToolStripMenuItem,
            this.blueToolStripMenuItem,
            this.redToolStripMenuItem,
            this.greenToolStripMenuItem,
            this.yellowToolStripMenuItem,
            this.whiteToolStripMenuItem});
            this.gridColorToolStripMenuItem.Name = "gridColorToolStripMenuItem";
            this.gridColorToolStripMenuItem.Size = new System.Drawing.Size(137, 22);
            this.gridColorToolStripMenuItem.Text = "&Grid Color";
            //
            // blackToolStripMenuItem
            //
            this.blackToolStripMenuItem.Name = "blackToolStripMenuItem";
            this.blackToolStripMenuItem.Size = new System.Drawing.Size(108, 22);
            this.blackToolStripMenuItem.Text = "&Black";
            //
            // blueToolStripMenuItem
            //
            this.blueToolStripMenuItem.Name = "blueToolStripMenuItem";
            this.blueToolStripMenuItem.Size = new System.Drawing.Size(108, 22);
            this.blueToolStripMenuItem.Text = "B&lue";
            //
            // redToolStripMenuItem
            //
            this.redToolStripMenuItem.Name = "redToolStripMenuItem";
            this.redToolStripMenuItem.Size = new System.Drawing.Size(108, 22);
            this.redToolStripMenuItem.Text = "R&ed";
            //
            // greenToolStripMenuItem
            //
            this.greenToolStripMenuItem.Name = "greenToolStripMenuItem";
            this.greenToolStripMenuItem.Size = new System.Drawing.Size(108, 22);
            this.greenToolStripMenuItem.Text = "&Green";
            //
            // yellowToolStripMenuItem
            //
            this.yellowToolStripMenuItem.Name = "yellowToolStripMenuItem";
            this.yellowToolStripMenuItem.Size = new System.Drawing.Size(108, 22);
            this.yellowToolStripMenuItem.Text = "&Yellow";
            //
            // whiteToolStripMenuItem
```

```
            //
            this.whiteToolStripMenuItem.Checked = true;
            this.whiteToolStripMenuItem.CheckState = System.Windows.Forms.CheckState.Checked;
            this.whiteToolStripMenuItem.Name = "whiteToolStripMenuItem";
            this.whiteToolStripMenuItem.Size = new System.Drawing.Size(108, 22);
            this.whiteToolStripMenuItem.Text = "&White";
            //
            // tilesetToolStripMenuItem
            //
            this.tilesetToolStripMenuItem.DropDownItems.AddRange(new
System.Windows.Forms.ToolStripItem[] {
            this.newTilesetToolStripMenuItem,
            this.openTilesetToolStripMenuItem,
            this.saveTilesetToolStripMenuItem,
            this.animatedTIlesetToolStripMenuItem});
            this.tilesetToolStripMenuItem.Name = "tilesetToolStripMenuItem";
            this.tilesetToolStripMenuItem.Size = new System.Drawing.Size(53, 20);
            this.tilesetToolStripMenuItem.Text = "&Tileset";
            //
            // newTilesetToolStripMenuItem
            //
            this.newTilesetToolStripMenuItem.Name = "newTilesetToolStripMenuItem";
            this.newTilesetToolStripMenuItem.Size = new System.Drawing.Size(163, 22);
            this.newTilesetToolStripMenuItem.Text = "&New Tileset";
            //
            // openTilesetToolStripMenuItem
            //
            this.openTilesetToolStripMenuItem.Name = "openTilesetToolStripMenuItem";
            this.openTilesetToolStripMenuItem.Size = new System.Drawing.Size(163, 22);
            this.openTilesetToolStripMenuItem.Text = "&Open Tileset";
            //
            // saveTilesetToolStripMenuItem
            //
            this.saveTilesetToolStripMenuItem.Name = "saveTilesetToolStripMenuItem";
            this.saveTilesetToolStripMenuItem.Size = new System.Drawing.Size(163, 22);
            this.saveTilesetToolStripMenuItem.Text = "&Save Tileset";
            //
            // mapLayerToolStripMenuItem
            //
            this.mapLayerToolStripMenuItem.DropDownItems.AddRange(new
System.Windows.Forms.ToolStripItem[] {
            this.newLayerToolStripMenuItem,
            this.openLayerToolStripMenuItem,
            this.saveLayerToolStripMenuItem});
            this.mapLayerToolStripMenuItem.Name = "mapLayerToolStripMenuItem";
            this.mapLayerToolStripMenuItem.Size = new System.Drawing.Size(74, 20);
            this.mapLayerToolStripMenuItem.Text = "&Map Layer";
            //
            // newLayerToolStripMenuItem
            //
            this.newLayerToolStripMenuItem.Name = "newLayerToolStripMenuItem";
            this.newLayerToolStripMenuItem.Size = new System.Drawing.Size(134, 22);
            this.newLayerToolStripMenuItem.Text = "&New Layer";
            //
            // openLayerToolStripMenuItem
            //
            this.openLayerToolStripMenuItem.Name = "openLayerToolStripMenuItem";
            this.openLayerToolStripMenuItem.Size = new System.Drawing.Size(134, 22);
            this.openLayerToolStripMenuItem.Text = "&Open Layer";
            //
```

```
            // saveLayerToolStripMenuItem
            //
            this.saveLayerToolStripMenuItem.Name = "saveLayerToolStripMenuItem";
            this.saveLayerToolStripMenuItem.Size = new System.Drawing.Size(134, 22);
            this.saveLayerToolStripMenuItem.Text = "&Save Layer";
            //
            // charactersToolStripMenuItem
            //
            this.charactersToolStripMenuItem.Name = "charactersToolStripMenuItem";
            this.charactersToolStripMenuItem.Size = new System.Drawing.Size(75, 20);
            this.charactersToolStripMenuItem.Text = "&Characters";
            //
            // chestsToolStripMenuItem
            //
            this.chestsToolStripMenuItem.Name = "chestsToolStripMenuItem";
            this.chestsToolStripMenuItem.Size = new System.Drawing.Size(54, 20);
            this.chestsToolStripMenuItem.Text = "C&hests";
            //
            // keysToolStripMenuItem
            //
            this.keysToolStripMenuItem.Name = "keysToolStripMenuItem";
            this.keysToolStripMenuItem.Size = new System.Drawing.Size(43, 20);
            this.keysToolStripMenuItem.Text = "&Keys";
            //
            // brushesToolStripMenuItem
            //
            this.brushesToolStripMenuItem.DropDownItems.AddRange(new
System.Windows.Forms.ToolStripItem[] {
            this.x1ToolStripMenuItem,
            this.x2ToolStripMenuItem,
            this.x4ToolStripMenuItem,
            this.x8ToolStripMenuItem});
            this.brushesToolStripMenuItem.Name = "brushesToolStripMenuItem";
            this.brushesToolStripMenuItem.Size = new System.Drawing.Size(60, 20);
            this.brushesToolStripMenuItem.Text = "&Brushes";
            //
            // x1ToolStripMenuItem
            //
            this.x1ToolStripMenuItem.Checked = true;
            this.x1ToolStripMenuItem.CheckState = System.Windows.Forms.CheckState.Checked;
            this.x1ToolStripMenuItem.Name = "x1ToolStripMenuItem";
            this.x1ToolStripMenuItem.Size = new System.Drawing.Size(97, 22);
            this.x1ToolStripMenuItem.Text = "1 x 1";
            //
            // x2ToolStripMenuItem
            //
            this.x2ToolStripMenuItem.Name = "x2ToolStripMenuItem";
            this.x2ToolStripMenuItem.Size = new System.Drawing.Size(97, 22);
            this.x2ToolStripMenuItem.Text = "2 x 2";
            //
            // x4ToolStripMenuItem
            //
            this.x4ToolStripMenuItem.Name = "x4ToolStripMenuItem";
            this.x4ToolStripMenuItem.Size = new System.Drawing.Size(97, 22);
            this.x4ToolStripMenuItem.Text = "4 x 4";
            //
            // x8ToolStripMenuItem
            //
            this.x8ToolStripMenuItem.Name = "x8ToolStripMenuItem";
            this.x8ToolStripMenuItem.Size = new System.Drawing.Size(97, 22);
```

```
this.x8ToolStripMenuItem.Text = "8 x 8";
//
// splitContainer1
//
this.splitContainer1.Dock = System.Windows.Forms.DockStyle.Fill;
this.splitContainer1.Location = new System.Drawing.Point(0, 24);
this.splitContainer1.Margin = new System.Windows.Forms.Padding(2);
this.splitContainer1.Name = "splitContainer1";
//
// splitContainer1.Panel1
//
this.splitContainer1.Panel1.Controls.Add(this.mapDisplay);
//
// splitContainer1.Panel2
//
this.splitContainer1.Panel2.Controls.Add(this.tabProperties);
this.splitContainer1.Size = new System.Drawing.Size(1151, 576);
this.splitContainer1.SplitterDistance = 914;
this.splitContainer1.SplitterWidth = 3;
this.splitContainer1.TabIndex = 1;
//
// tabProperties
//
this.tabProperties.Controls.Add(this.tabTilesets);
this.tabProperties.Controls.Add(this.tabLayers);
this.tabProperties.Controls.Add(this.tabAnimated);
this.tabProperties.Controls.Add(this.tabCollisions);
this.tabProperties.Controls.Add(this.tabCharacters);
this.tabProperties.Controls.Add(this.tabChests);
this.tabProperties.Controls.Add(this.tabKeys);
this.tabProperties.Dock = System.Windows.Forms.DockStyle.Fill;
this.tabProperties.Location = new System.Drawing.Point(0, 0);
this.tabProperties.Margin = new System.Windows.Forms.Padding(2);
this.tabProperties.Name = "tabProperties";
this.tabProperties.SelectedIndex = 0;
this.tabProperties.Size = new System.Drawing.Size(234, 576);
this.tabProperties.TabIndex = 1;
//
// tabTilesets
//
this.tabTilesets.Controls.Add(this.tbMapLocation);
this.tabTilesets.Controls.Add(this.lblCursor);
this.tabTilesets.Controls.Add(this.lblTilesets);
this.tabTilesets.Controls.Add(this.lbTileset);
this.tabTilesets.Controls.Add(this.pbTilesetPreview);
this.tabTilesets.Controls.Add(this.lblCurrentTileset);
this.tabTilesets.Controls.Add(this.nudCurrentTile);
this.tabTilesets.Controls.Add(this.gbDrawMode);
this.tabTilesets.Controls.Add(this.lblTile);
this.tabTilesets.Controls.Add(this.pbTilePreview);
this.tabTilesets.Location = new System.Drawing.Point(4, 22);
this.tabTilesets.Margin = new System.Windows.Forms.Padding(2);
this.tabTilesets.Name = "tabTilesets";
this.tabTilesets.Padding = new System.Windows.Forms.Padding(2);
this.tabTilesets.Size = new System.Drawing.Size(226, 550);
this.tabTilesets.TabIndex = 0;
this.tabTilesets.Text = "Tiles";
this.tabTilesets.UseVisualStyleBackColor = true;
//
// tbMapLocation
```

```
            //
            this.tbMapLocation.Location = new System.Drawing.Point(6, 479);
            this.tbMapLocation.Margin = new System.Windows.Forms.Padding(2);
            this.tbMapLocation.Name = "tbMapLocation";
            this.tbMapLocation.Size = new System.Drawing.Size(136, 20);
            this.tbMapLocation.TabIndex = 9;
            //
            // lblCursor
            //
            this.lblCursor.Location = new System.Drawing.Point(6, 461);
            this.lblCursor.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);
            this.lblCursor.Name = "lblCursor";
            this.lblCursor.Size = new System.Drawing.Size(135, 19);
            this.lblCursor.TabIndex = 8;
            this.lblCursor.Text = "Map Location";
            this.lblCursor.TextAlign = System.Drawing.ContentAlignment.TopCenter;
            //
            // lblTilesets
            //
            this.lblTilesets.Location = new System.Drawing.Point(5, 261);
            this.lblTilesets.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);
            this.lblTilesets.Name = "lblTilesets";
            this.lblTilesets.Size = new System.Drawing.Size(135, 19);
            this.lblTilesets.TabIndex = 7;
            this.lblTilesets.Text = "Tilesets";
            this.lblTilesets.TextAlign = System.Drawing.ContentAlignment.TopCenter;
            //
            // lbTileset
            //
            this.lbTileset.FormattingEnabled = true;
            this.lbTileset.Location = new System.Drawing.Point(5, 286);
            this.lbTileset.Margin = new System.Windows.Forms.Padding(2);
            this.lbTileset.Name = "lbTileset";
            this.lbTileset.Size = new System.Drawing.Size(136, 173);
            this.lbTileset.TabIndex = 6;
            //
            // pbTilesetPreview
            //
            this.pbTilesetPreview.Location = new System.Drawing.Point(5, 112);
            this.pbTilesetPreview.Margin = new System.Windows.Forms.Padding(2);
            this.pbTilesetPreview.Name = "pbTilesetPreview";
            this.pbTilesetPreview.Size = new System.Drawing.Size(135, 146);
            this.pbTilesetPreview.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pbTilesetPreview.TabIndex = 5;
            //
            // lblCurrentTileset
            //
            this.lblCurrentTileset.Location = new System.Drawing.Point(5, 91);
            this.lblCurrentTileset.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);
            this.lblCurrentTileset.Name = "lblCurrentTileset";
            this.lblCurrentTileset.Size = new System.Drawing.Size(135, 19);
            this.lblCurrentTileset.TabIndex = 4;
            this.lblCurrentTileset.Text = "Current Tileset";
            this.lblCurrentTileset.TextAlign = System.Drawing.ContentAlignment.TopCenter;
            //
            // nudCurrentTile
            //
            this.nudCurrentTile.Location = new System.Drawing.Point(5, 67);
            this.nudCurrentTile.Margin = new System.Windows.Forms.Padding(2);
```

```
this.nudCurrentTile.Name = "nudCurrentTile";
this.nudCurrentTile.Size = new System.Drawing.Size(135, 20);
this.nudCurrentTile.TabIndex = 3;
//
// gbDrawMode
//
this.gbDrawMode.Controls.Add(this.rbErase);
this.gbDrawMode.Controls.Add(this.rbDraw);
this.gbDrawMode.Location = new System.Drawing.Point(47, 6);
this.gbDrawMode.Margin = new System.Windows.Forms.Padding(2);
this.gbDrawMode.Name = "gbDrawMode";
this.gbDrawMode.Padding = new System.Windows.Forms.Padding(2);
this.gbDrawMode.Size = new System.Drawing.Size(96, 57);
this.gbDrawMode.TabIndex = 2;
this.gbDrawMode.TabStop = false;
this.gbDrawMode.Text = "Draw Mode";
//
// rbErase
//
this.rbErase.AutoSize = true;
this.rbErase.Location = new System.Drawing.Point(5, 35);
this.rbErase.Margin = new System.Windows.Forms.Padding(2);
this.rbErase.Name = "rbErase";
this.rbErase.Size = new System.Drawing.Size(52, 17);
this.rbErase.TabIndex = 1;
this.rbErase.Text = "Erase";
this.rbErase.UseVisualStyleBackColor = true;
//
// rbDraw
//
this.rbDraw.AutoSize = true;
this.rbDraw.Checked = true;
this.rbDraw.Location = new System.Drawing.Point(5, 16);
this.rbDraw.Margin = new System.Windows.Forms.Padding(2);
this.rbDraw.Name = "rbDraw";
this.rbDraw.Size = new System.Drawing.Size(50, 17);
this.rbDraw.TabIndex = 0;
this.rbDraw.TabStop = true;
this.rbDraw.Text = "Draw";
this.rbDraw.UseVisualStyleBackColor = true;
//
// lblTile
//
this.lblTile.Location = new System.Drawing.Point(5, 6);
this.lblTile.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);
this.lblTile.Name = "lblTile";
this.lblTile.Size = new System.Drawing.Size(38, 14);
this.lblTile.TabIndex = 1;
this.lblTile.Text = "Tile";
this.lblTile.TextAlign = System.Drawing.ContentAlignment.TopCenter;
//
// pbTilePreview
//
this.pbTilePreview.Location = new System.Drawing.Point(5, 22);
this.pbTilePreview.Margin = new System.Windows.Forms.Padding(2);
this.pbTilePreview.Name = "pbTilePreview";
this.pbTilePreview.Size = new System.Drawing.Size(38, 41);
this.pbTilePreview.TabIndex = 0;
this.pbTilePreview.TabStop = false;
//
```

```
// tabLayers
//
this.tabLayers.Controls.Add(this.clbLayers);
this.tabLayers.Location = new System.Drawing.Point(4, 22);
this.tabLayers.Margin = new System.Windows.Forms.Padding(2);
this.tabLayers.Name = "tabLayers";
this.tabLayers.Padding = new System.Windows.Forms.Padding(2);
this.tabLayers.Size = new System.Drawing.Size(226, 550);
this.tabLayers.TabIndex = 1;
this.tabLayers.Text = "Map Layers";
this.tabLayers.UseVisualStyleBackColor = true;
//
// clbLayers
//
this.clbLayers.Dock = System.Windows.Forms.DockStyle.Fill;
this.clbLayers.FormattingEnabled = true;
this.clbLayers.Location = new System.Drawing.Point(2, 2);
this.clbLayers.Margin = new System.Windows.Forms.Padding(2);
this.clbLayers.Name = "clbLayers";
this.clbLayers.Size = new System.Drawing.Size(222, 546);
this.clbLayers.TabIndex = 0;
//
// tabCharacters
//
this.tabCharacters.Location = new System.Drawing.Point(4, 22);
this.tabCharacters.Margin = new System.Windows.Forms.Padding(2);
this.tabCharacters.Name = "tabCharacters";
this.tabCharacters.Size = new System.Drawing.Size(226, 550);
this.tabCharacters.TabIndex = 2;
this.tabCharacters.Text = "Characters";
this.tabCharacters.UseVisualStyleBackColor = true;
//
// tabChests
//
this.tabChests.Location = new System.Drawing.Point(4, 22);
this.tabChests.Margin = new System.Windows.Forms.Padding(2);
this.tabChests.Name = "tabChests";
this.tabChests.Size = new System.Drawing.Size(226, 550);
this.tabChests.TabIndex = 3;
this.tabChests.Text = "Chests";
this.tabChests.UseVisualStyleBackColor = true;
//
// tabKeys
//
this.tabKeys.Location = new System.Drawing.Point(4, 22);
this.tabKeys.Margin = new System.Windows.Forms.Padding(2);
this.tabKeys.Name = "tabKeys";
this.tabKeys.Size = new System.Drawing.Size(226, 550);
this.tabKeys.TabIndex = 4;
this.tabKeys.Text = "Keys";
this.tabKeys.UseVisualStyleBackColor = true;
//
// animatedTIlesetToolStripMenuItem
//
this.animatedTIlesetToolStripMenuItem.Name = "animatedTIlesetToolStripMenuItem";
this.animatedTIlesetToolStripMenuItem.Size = new System.Drawing.Size(163, 22);
this.animatedTIlesetToolStripMenuItem.Text = "&Animated TIleset";
//
// tabAnimated
//
```

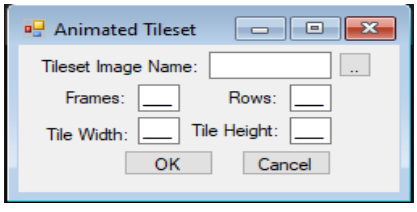
```
            this.tabAnimated.Controls.Add(this.sbAnimatedTile);
            this.tabAnimated.Controls.Add(this.pbAnimatedSet);
            this.tabAnimated.Controls.Add(this.chkPaintAnimated);
            this.tabAnimated.Location = new System.Drawing.Point(4, 22);
            this.tabAnimated.Name = "tabAnimated";
            this.tabAnimated.Padding = new System.Windows.Forms.Padding(3);
            this.tabAnimated.Size = new System.Drawing.Size(226, 550);
            this.tabAnimated.TabIndex = 5;
            this.tabAnimated.Text = "Animated Tiles";
            this.tabAnimated.UseVisualStyleBackColor = true;
            //
            // tabCollisions
            //
            this.tabCollisions.Controls.Add(this.groupBox1);
            this.tabCollisions.Controls.Add(this.chkPaintCollision);
            this.tabCollisions.Location = new System.Drawing.Point(4, 22);
            this.tabCollisions.Name = "tabCollisions";
            this.tabCollisions.Size = new System.Drawing.Size(226, 550);
            this.tabCollisions.TabIndex = 6;
            this.tabCollisions.Text = "Collision Layer";
            this.tabCollisions.UseVisualStyleBackColor = true;
            //
            // chkPaintAnimated
            //
            this.chkPaintAnimated.AutoSize = true;
            this.chkPaintAnimated.Location = new System.Drawing.Point(6, 6);
            this.chkPaintAnimated.Name = "chkPaintAnimated";
            this.chkPaintAnimated.Size = new System.Drawing.Size(112, 17);
            this.chkPaintAnimated.TabIndex = 0;
            this.chkPaintAnimated.Text = "Paint animated tile";
            this.chkPaintAnimated.UseVisualStyleBackColor = true;
            //
            // pbAnimatedSet
            //
            this.pbAnimatedSet.Anchor = ((System.Windows.Forms.AnchorStyles)
((((System.Windows.Forms.AnchorStyles.Top | System.Windows.Forms.AnchorStyles.Bottom)
                        | System.Windows.Forms.AnchorStyles.Left)
                        | System.Windows.Forms.AnchorStyles.Right)));
            this.pbAnimatedSet.Location = new System.Drawing.Point(0, 55);
            this.pbAnimatedSet.Name = "pbAnimatedSet";
            this.pbAnimatedSet.Size = new System.Drawing.Size(227, 492);
            this.pbAnimatedSet.SizeMode = System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pbAnimatedSet.TabIndex = 1;
            this.pbAnimatedSet.TabStop = false;
            //
            // chkPaintCollision
            //
            this.chkPaintCollision.AutoSize = true;
            this.chkPaintCollision.Location = new System.Drawing.Point(3, 3);
            this.chkPaintCollision.Name = "chkPaintCollision";
            this.chkPaintCollision.Size = new System.Drawing.Size(90, 17);
            this.chkPaintCollision.TabIndex = 0;
            this.chkPaintCollision.Text = "Paint collision";
            this.chkPaintCollision.UseVisualStyleBackColor = true;
            //
            // groupBox1
            //
            this.groupBox1.Controls.Add(this.rbImpassable);
            this.groupBox1.Controls.Add(this.rbNoCollision);
            this.groupBox1.Location = new System.Drawing.Point(3, 26);
```

```
this.groupBox1.Name = "groupBox1";
this.groupBox1.Size = new System.Drawing.Size(215, 144);
this.groupBox1.TabIndex = 1;
this.groupBox1.TabStop = false;
this.groupBox1.Text = "Collision types";
//
// rbNoCollision
//
this.rbNoCollision.AutoSize = true;
this.rbNoCollision.Checked = true;
this.rbNoCollision.Location = new System.Drawing.Point(6, 19);
this.rbNoCollision.Name = "rbNoCollision";
this.rbNoCollision.Size = new System.Drawing.Size(80, 17);
this.rbNoCollision.TabIndex = 0;
this.rbNoCollision.TabStop = true;
this.rbNoCollision.Text = "No Collision";
this.rbNoCollision.UseVisualStyleBackColor = true;
//
// rbImpassable
//
this.rbImpassable.AutoSize = true;
this.rbImpassable.Location = new System.Drawing.Point(6, 42);
this.rbImpassable.Name = "rbImpassable";
this.rbImpassable.Size = new System.Drawing.Size(78, 17);
this.rbImpassable.TabIndex = 1;
this.rbImpassable.Text = "Impassable";
this.rbImpassable.UseVisualStyleBackColor = true;
//
// mapDisplay
//
this.mapDisplay.Dock = System.Windows.Forms.DockStyle.Fill;
this.mapDisplay.Location = new System.Drawing.Point(0, 0);
this.mapDisplay.Margin = new System.Windows.Forms.Padding(2);
this.mapDisplay.Name = "mapDisplay";
this.mapDisplay.Size = new System.Drawing.Size(914, 576);
this.mapDisplay.TabIndex = 0;
this.mapDisplay.Text = "mapDisplay1";
//
// sbAnimatedTile
//
this.sbAnimatedTile.Location = new System.Drawing.Point(6, 29);
this.sbAnimatedTile.Maximum = new decimal(new int[] {
1,
0,
0,
-2147483648});
this.sbAnimatedTile.Minimum = new decimal(new int[] {
1,
0,
0,
-2147483648});
this.sbAnimatedTile.Name = "sbAnimatedTile";
this.sbAnimatedTile.Size = new System.Drawing.Size(120, 20);
this.sbAnimatedTile.TabIndex = 2;
this.sbAnimatedTile.Value = new decimal(new int[] {
1,
0,
0,
-2147483648});
//
```

```
        // FormMain
        //
        this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
        this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
        this.ClientSize = new System.Drawing.Size(1151, 600);
        this.Controls.Add(this.splitContainer1);
        this.Controls.Add(this.menuStrip1);
        this.MainMenuStrip = this.menuStrip1;
        this.Margin = new System.Windows.Forms.Padding(2);
        this.Name = "FormMain";
        this.StartPosition = System.Windows.Forms.FormStartPosition.CenterParent;
        this.Text = "Level Editor";
        this.menuStrip1.ResumeLayout(false);
        this.menuStrip1.PerformLayout();
        this.splitContainer1.Panel1.ResumeLayout(false);
        this.splitContainer1.Panel2.ResumeLayout(false);
        ((System.ComponentModel.ISupportInitialize)(this.splitContainer1)).EndInit();
        this.splitContainer1.ResumeLayout(false);
        this.tabProperties.ResumeLayout(false);
        this.tabTilesets.ResumeLayout(false);
        this.tabTilesets.PerformLayout();
        ((System.ComponentModel.ISupportInitialize)(this.pbTilesetPreview)).EndInit();
        ((System.ComponentModel.ISupportInitialize)(this.nudCurrentTile)).EndInit();
        this.gbDrawMode.ResumeLayout(false);
        this.gbDrawMode.PerformLayout();
        ((System.ComponentModel.ISupportInitialize)(this.pbTilePreview)).EndInit();
        this.tabLayers.ResumeLayout(false);
        this.tabAnimated.ResumeLayout(false);
        this.tabAnimated.PerformLayout();
        this.tabCollisions.ResumeLayout(false);
        this.tabCollisions.PerformLayout();
        ((System.ComponentModel.ISupportInitialize)(this.pbAnimatedSet)).EndInit();
        this.groupBox1.ResumeLayout(false);
        this.groupBox1.PerformLayout();
        ((System.ComponentModel.ISupportInitialize)(this.sbAnimatedTile)).EndInit();
        this.ResumeLayout(false);
        this.PerformLayout();

    }

    #endregion

    private System.Windows.Forms.MenuStrip menuStrip1;
    private System.Windows.Forms.ToolStripMenuItem levelToolStripMenuItem;
    private System.Windows.Forms.ToolStripMenuItem newLevelToolStripMenuItem;
    private System.Windows.Forms.ToolStripMenuItem openLevelToolStripMenuItem;
    private System.Windows.Forms.ToolStripMenuItem saveLevelToolStripMenuItem;
    private System.Windows.Forms.ToolStripSeparator toolStripMenuItem1;
    private System.Windows.Forms.ToolStripMenuItem exitEditorToolStripMenuItem;
    private System.Windows.Forms.ToolStripMenuItem tilesetToolStripMenuItem;
    private System.Windows.Forms.ToolStripMenuItem newTilesetToolStripMenuItem;
    private System.Windows.Forms.ToolStripMenuItem openTilesetToolStripMenuItem;
    private System.Windows.Forms.ToolStripMenuItem mapLayerToolStripMenuItem;
    private System.Windows.Forms.ToolStripMenuItem newLayerToolStripMenuItem;
    private System.Windows.Forms.ToolStripMenuItem openLayerToolStripMenuItem;
    private System.Windows.Forms.ToolStripMenuItem saveLayerToolStripMenuItem;
    private System.Windows.Forms.ToolStripMenuItem charactersToolStripMenuItem;
    private System.Windows.Forms.ToolStripMenuItem chestsToolStripMenuItem;
    private System.Windows.Forms.ToolStripMenuItem keysToolStripMenuItem;
    private System.Windows.Forms.SplitContainer splitContainer1;
```

```
        private MapDisplay mapDisplay;
        private System.Windows.Forms.TabControl tabProperties;
        private System.Windows.Forms.TabPage tabTilesets;
        private System.Windows.Forms.TabPage tabLayers;
        private System.Windows.Forms.TabPage tabCharacters;
        private System.Windows.Forms.TabPage tabChests;
        private System.Windows.Forms.TabPage tabKeys;
        private System.Windows.Forms.Label lblTilesets;
        private System.Windows.Forms.ListBox lbTileset;
        private System.Windows.Forms.PictureBox pbTilesetPreview;
        private System.Windows.Forms.Label lblCurrentTileset;
        private System.Windows.Forms.NumericUpDown nudCurrentTile;
        private System.Windows.Forms.GroupBox gbDrawMode;
        private System.Windows.Forms.RadioButton rbErase;
        private System.Windows.Forms.RadioButton rbDraw;
        private System.Windows.Forms.Label lblTile;
        private System.Windows.Forms.PictureBox pbTilePreview;
        private System.Windows.Forms.CheckedListBox clbLayers;
        private System.Windows.Forms.Timer controlTimer;
        private System.Windows.Forms.TextBox tbMapLocation;
        private System.Windows.Forms.Label lblCursor;
        private System.Windows.Forms.ToolStripMenuItem viewToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem displayGridToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem brushesToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem x1ToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem x2ToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem x4ToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem x8ToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem gridColorToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem blackToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem blueToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem redToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem greenToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem yellowToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem whiteToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem saveTilesetToolStripMenuItem;
        private System.Windows.Forms.ToolStripMenuItem animatedTIlesetToolStripMenuItem;
        private System.Windows.Forms.TabPage tabAnimated;
        private System.Windows.Forms.TabPage tabCollisions;
        private System.Windows.Forms.PictureBox pbAnimatedSet;
        private System.Windows.Forms.CheckBox chkPaintAnimated;
        private System.Windows.Forms.GroupBox groupBox1;
        private System.Windows.Forms.RadioButton rbImpassable;
        private System.Windows.Forms.RadioButton rbNoCollision;
        private System.Windows.Forms.CheckBox chkPaintCollision;
        private System.Windows.Forms.NumericUpDown sbAnimatedTile;
    }
}
```

I need to add a new form to the project to create an animated tile set and add it to the map. Right click the XlevelEditor project, select Add and Form (Windows Form). Name this new form FormNewAnimatedTileset. My finished form looked like this.



Alternatively to designing the form manually you can copy and paste the code for the designer that follows after the instructions. Set the Text property of the form to Animated Tileset and set the Text properties of the labels as seen in the image. For the text box for the image name set the (Name) property to tbTilesetImage and the Enabled property to False. For the top button set the (Name) property to btnSelectImage and the Text property to .... The other four boxes are MaskedTextBoxes. What I did is created the first one and setits properties. I then copied and pasted the other three. All of them have a the Mask property 000. Their (Name) properties are tbFrames, tbRows, tbTileWidth and tbTileHeight. The OK button has (Name) btnOK and Text OK. The Cancel button has (Name) btnCancel and Text Cancel.

Here is the code for the FormNewAnimatedTileset.Designer.cs file.

```
namespace XLevelEditor
{
    partial class FormNewAnimatedTileset
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.btnCancel = new System.Windows.Forms.Button();
            this.btnOK = new System.Windows.Forms.Button();
            this.tbTileHeight = new System.Windows.Forms.MaskedTextBox();
            this.lblTileHeight = new System.Windows.Forms.Label();
            this.tbTileWidth = new System.Windows.Forms.MaskedTextBox();
            this.lblTileWidth = new System.Windows.Forms.Label();
            this.btnSelectImage = new System.Windows.Forms.Button();
            this.tbTilesetImage = new System.Windows.Forms.TextBox();
            this.lblTilesetImageName = new System.Windows.Forms.Label();
            this.tbRows = new System.Windows.Forms.MaskedTextBox();
            this.label1 = new System.Windows.Forms.Label();
            this.tbFrames = new System.Windows.Forms.MaskedTextBox();
            this.label2 = new System.Windows.Forms.Label();
            this.SuspendLayout();
```

```
//
// btnCancel
//
this.btnCancel.Location = new System.Drawing.Point(138, 77);
this.btnCancel.Margin = new System.Windows.Forms.Padding(2);
this.btnCancel.Name = "btnCancel";
this.btnCancel.Size = new System.Drawing.Size(56, 19);
this.btnCancel.TabIndex = 25;
this.btnCancel.Text = "Cancel";
this.btnCancel.UseVisualStyleBackColor = true;
//
// btnOK
//
this.btnOK.Location = new System.Drawing.Point(65, 77);
this.btnOK.Margin = new System.Windows.Forms.Padding(2);
this.btnOK.Name = "btnOK";
this.btnOK.Size = new System.Drawing.Size(56, 19);
this.btnOK.TabIndex = 24;
this.btnOK.Text = "OK";
this.btnOK.UseVisualStyleBackColor = true;
//
// tbTileHeight
//
this.tbTileHeight.Location = new System.Drawing.Point(168, 53);
this.tbTileHeight.Margin = new System.Windows.Forms.Padding(2);
this.tbTileHeight.Mask = "000";
this.tbTileHeight.Name = "tbTileHeight";
this.tbTileHeight.Size = new System.Drawing.Size(26, 20);
this.tbTileHeight.TabIndex = 23;
//
// lblTileHeight
//
this.lblTileHeight.AutoSize = true;
this.lblTileHeight.Location = new System.Drawing.Point(104, 56);
this.lblTileHeight.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);
this.lblTileHeight.Name = "lblTileHeight";
this.lblTileHeight.Size = new System.Drawing.Size(61, 13);
this.lblTileHeight.TabIndex = 22;
this.lblTileHeight.Text = "Tile Height:";
//
// tbTileWidth
//
this.tbTileWidth.Location = new System.Drawing.Point(74, 53);
this.tbTileWidth.Margin = new System.Windows.Forms.Padding(2);
this.tbTileWidth.Mask = "000";
this.tbTileWidth.Name = "tbTileWidth";
this.tbTileWidth.Size = new System.Drawing.Size(26, 20);
this.tbTileWidth.TabIndex = 21;
//
// lblTileWidth
//
this.lblTileWidth.AutoSize = true;
this.lblTileWidth.Location = new System.Drawing.Point(14, 59);
this.lblTileWidth.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);
this.lblTileWidth.Name = "lblTileWidth";
this.lblTileWidth.Size = new System.Drawing.Size(58, 13);
this.lblTileWidth.TabIndex = 20;
this.lblTileWidth.Text = "Tile Width:";
//
// btnSelectImage
```

```
            //
            this.btnSelectImage.Location = new System.Drawing.Point(198, 5);
            this.btnSelectImage.Margin = new System.Windows.Forms.Padding(2);
            this.btnSelectImage.Name = "btnSelectImage";
            this.btnSelectImage.Size = new System.Drawing.Size(21, 19);
            this.btnSelectImage.TabIndex = 19;
            this.btnSelectImage.Tag = "";
            this.btnSelectImage.Text = "...";
            this.btnSelectImage.UseVisualStyleBackColor = true;
            //
            // tbTilesetImage
            //
            this.tbTilesetImage.Enabled = false;
            this.tbTilesetImage.Location = new System.Drawing.Point(118, 5);
            this.tbTilesetImage.Margin = new System.Windows.Forms.Padding(2);
            this.tbTilesetImage.Name = "tbTilesetImage";
            this.tbTilesetImage.Size = new System.Drawing.Size(76, 20);
            this.tbTilesetImage.TabIndex = 18;
            //
            // lblTilesetImageName
            //
            this.lblTilesetImageName.AutoSize = true;
            this.lblTilesetImageName.Location = new System.Drawing.Point(11, 9);
            this.lblTilesetImageName.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);
            this.lblTilesetImageName.Name = "lblTilesetImageName";
            this.lblTilesetImageName.Size = new System.Drawing.Size(104, 13);
            this.lblTilesetImageName.TabIndex = 17;
            this.lblTilesetImageName.Text = "Tileset Image Name:";
            //
            // tbRows
            //
            this.tbRows.Location = new System.Drawing.Point(168, 29);
            this.tbRows.Margin = new System.Windows.Forms.Padding(2);
            this.tbRows.Mask = "000";
            this.tbRows.Name = "tbRows";
            this.tbRows.Size = new System.Drawing.Size(26, 20);
            this.tbRows.TabIndex = 29;
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(127, 32);
            this.label1.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(37, 13);
            this.label1.TabIndex = 28;
            this.label1.Text = "Rows:";
            //
            // tbFrames
            //
            this.tbFrames.Location = new System.Drawing.Point(74, 29);
            this.tbFrames.Margin = new System.Windows.Forms.Padding(2);
            this.tbFrames.Mask = "000";
            this.tbFrames.Name = "tbFrames";
            this.tbFrames.Size = new System.Drawing.Size(26, 20);
            this.tbFrames.TabIndex = 27;
            //
            // label2
            //
            this.label2.AutoSize = true;
```

```
            this.label2.Location = new System.Drawing.Point(26, 32);
            this.label2.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(44, 13);
            this.label2.TabIndex = 26;
            this.label2.Text = "Frames:";
            //
            // FormNewAnimatedTileset
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(236, 108);
            this.Controls.Add(this.tbRows);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.tbFrames);
            this.Controls.Add(this.label2);
            this.Controls.Add(this.btnCancel);
            this.Controls.Add(this.btnOK);
            this.Controls.Add(this.tbTileHeight);
            this.Controls.Add(this.lblTileHeight);
            this.Controls.Add(this.tbTileWidth);
            this.Controls.Add(this.lblTileWidth);
            this.Controls.Add(this.btnSelectImage);
            this.Controls.Add(this.tbTilesetImage);
            this.Controls.Add(this.lblTilesetImageName);
            this.Name = "FormNewAnimatedTileset";
            this.Text = "Animated Tileset";
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.Button btnCancel;
        private System.Windows.Forms.Button btnOK;
        private System.Windows.Forms.MaskedTextBox tbTileHeight;
        private System.Windows.Forms.Label lblTileHeight;
        private System.Windows.Forms.MaskedTextBox tbTileWidth;
        private System.Windows.Forms.Label lblTileWidth;
        private System.Windows.Forms.Button btnSelectImage;
        private System.Windows.Forms.TextBox tbTilesetImage;
        private System.Windows.Forms.Label lblTilesetImageName;
        private System.Windows.Forms.MaskedTextBox tbRows;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.MaskedTextBox tbFrames;
        private System.Windows.Forms.Label label2;
    }
}
```

Open the code for this form and replace it with the following.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
```

```
using RpgLibrary.WorldClasses;

namespace XLevelEditor
{
    public partial class FormNewAnimatedTileset : Form
    {
        public AnimatedTilesetData Tileset { get; set; }

        public FormNewAnimatedTileset()
        {
            InitializeComponent();

            btnSelectImage.Click += new EventHandler(btnSelectImage_Click);
            btnOK.Click += new EventHandler(btnOK_Click);
            btnCancel.Click += new EventHandler(btnCancel_Click);
        }

        void btnSelectImage_Click(object sender, EventArgs e)
        {
            OpenFileDialog ofDialog = new OpenFileDialog
            {
                Filter = "Image Files|*.BMP;*.GIF;*.JPG;*.TGA;*.PNG",
                CheckFileExists = true,
                CheckPathExists = true,
                Multiselect = false
            };

            DialogResult result = ofDialog.ShowDialog();

            if (result == DialogResult.OK)
            {
                tbTilesetImage.Text = ofDialog.FileName;
            }
        }

        void btnOK_Click(object sender, EventArgs e)
        {
            if (string.IsNullOrEmpty(tbTilesetImage.Text))
            {
                MessageBox.Show("You must select an image for the tileset in order to
proceed.");
                return;
            }

            int tileWidth = 0;
            int tileHeight = 0;
            int frames = 0;
            int rows = 0;

            if (!int.TryParse(tbTileWidth.Text, out tileWidth))
            {
                MessageBox.Show("Tile width must be an integer value.");
                return;
            }
            else if (tileWidth < 1)
            {
                MessageBox.Show("Tile width must me greater than zero.");
                return;
            }
```

```
            if (!int.TryParse(tbTileHeight.Text, out tileHeight))
            {
                MessageBox.Show("Tile height must be an integer value.");
                return;
            }
            else if (tileHeight < 1)
            {
                MessageBox.Show("Tile height must be greater than zero.");
                return;
            }

            if (!int.TryParse(tbFrames.Text, out frames))
            {
                MessageBox.Show("Frames must be an integer value.");
                return;
            }
            else if (frames < 1)
            {
                MessageBox.Show("Frames must me greater than zero.");
                return;
            }

            if (!int.TryParse(tbRows.Text, out rows))
            {
                MessageBox.Show("Rows must be an integer value.");
                return;
            }
            else if (rows < 1)
            {
                MessageBox.Show("Rows must be greater than zero.");
                return;
            }

            Tileset = new AnimatedTilesetData
            {
                TilesetName = tbTilesetImage.Text,
                TilesetImageName = tbTilesetImage.Text,
                FramesAcross = frames,
                TilesHigh = rows,
                TileWidthInPixels = tileWidth,
                TileHeightInPixels = tileHeight
            };

            this.Close();
        }

        void btnCancel_Click(object sender, EventArgs e)
        {
            Tileset = null;
            this.Close();
        }
    }
}
```

This is a very straight forward code behind for the form. There is a property that exposes an AnimatedTilesetData to be returned to the main form. The constructor wires the event handlers for the buttons. The event handler for the button to choose the image displays an OpenFileDialog to allow the user to select the image. If an image is selected the text property of the text box for the name is set to that value. The event handler for the OK button does some basic validation on the form to make

sure all values have been filled out with appropriate values. If one of them isn't it displays a message and exits the handler. Once everything is validated I create a new AnimatedTilesetData using the values and assign it to the property to return the value back to the main form and then closes the form. The cancel button sets the AniamtedTilesetData property to null and closes the form.
I need to add a method to the TileMap class that allows me to set the AnimatedTileset for the map because the only way to set it is through the constructor. You could also make it public or expose it as get and set through a property. Add this method to the TileMap class.

```
public void AddAnimatedTileset(AnimatedTileset tileset)
{
    animatedSet = tileset;
}
```

The next thing that I will do is update the constructor to wire some event handlers in the main form. What I want to do is add a handler for the new menu item and the check event of the check boxes for painting collisions or animated tiles. I also set them to be disabled, temporarily, until it makes sense to be able to use them. They are also mutually exclusive. If the collision one is selected the animated one should be deselected, and the same in reverse. Update the constructor for FormMain and add these event handlers.

```
public FormMain()
{
    InitializeComponent();

    this.Load += new EventHandler(FormMain_Load);
    this.FormClosing += new FormClosingEventHandler(FormMain_FormClosing);

    tilesetToolStripMenuItem.Enabled = false;
    mapLayerToolStripMenuItem.Enabled = false;
    charactersToolStripMenuItem.Enabled = false;
    chestsToolStripMenuItem.Enabled = false;
    keysToolStripMenuItem.Enabled = false;
    chkPaintAnimated.Enabled = false;
    chkPaintCollision.Enabled = false;

    newLevelToolStripMenuItem.Click += new EventHandler(newLevelToolStripMenuItem_Click);
    newTilesetToolStripMenuItem.Click += new EventHandler(newTilesetToolStripMenuItem_Click);
    newLayerToolStripMenuItem.Click += new EventHandler(newLayerToolStripMenuItem_Click);
    saveLevelToolStripMenuItem.Click += new EventHandler(saveLevelToolStripMenuItem_Click);
    openLevelToolStripMenuItem.Click += new EventHandler(openLevelToolStripMenuItem_Click);

    x1ToolStripMenuItem.Click += new EventHandler(x1ToolStripMenuItem_Click);
    x2ToolStripMenuItem.Click += new EventHandler(x2ToolStripMenuItem_Click);
    x4ToolStripMenuItem.Click += new EventHandler(x4ToolStripMenuItem_Click);
    x8ToolStripMenuItem.Click += new EventHandler(x8ToolStripMenuItem_Click);

    blackToolStripMenuItem.Click += new EventHandler(blackToolStripMenuItem_Click);
    blueToolStripMenuItem.Click += new EventHandler(blueToolStripMenuItem_Click);
    redToolStripMenuItem.Click += new EventHandler(redToolStripMenuItem_Click);
    greenToolStripMenuItem.Click += new EventHandler(greenToolStripMenuItem_Click);
    yellowToolStripMenuItem.Click += new EventHandler(yellowToolStripMenuItem_Click);
    whiteToolStripMenuItem.Click += new EventHandler(whiteToolStripMenuItem_Click);

    saveTilesetToolStripMenuItem.Click += new EventHandler(saveTilesetToolStripMenuItem_Click);
    saveLayerToolStripMenuItem.Click += new EventHandler(saveLayerToolStripMenuItem_Click);

    openTilesetToolStripMenuItem.Click += new EventHandler(openTilesetToolStripMenuItem_Click);
```

```
    openLayerToolStripMenuItem.Click += new EventHandler(openLayerToolStripMenuItem_Click);

    animatedTIlesetToolStripMenuItem.Click += new
EventHandler(animatedTIlesetToolStripMenuItem_Click);

    chkPaintAnimated.CheckedChanged += new EventHandler(chkPaintAnimated_CheckChanged);
    chkPaintCollision.CheckedChanged += new EventHandler(chkPaintCollision_CheckChanged);
}

private void animatedTIlesetToolStripMenuItem_Click(object sender, EventArgs e)
{
    FormNewAnimatedTileset frm = new FormNewAnimatedTileset();

    frm.ShowDialog();

    if (frm.Tileset == null)
    {
        return;
    }

    animatedSetData = frm.Tileset;

    try
    {
        GDIImage image = (GDIImage)GDIBitmap.FromFile(animatedSetData.TilesetImageName);

        pbAnimatedSet.Image = image;

        Stream stream = new FileStream(animatedSetData.TilesetImageName, FileMode.Open,
            FileAccess.Read);

        Texture2D texture = Texture2D.FromStream(GraphicsDevice, stream);
        animatedSet = new AnimatedTileset(
            texture,
            animatedSetData.FramesAcross,
            animatedSetData.TilesHigh,
            animatedSetData.TileWidthInPixels,
            animatedSetData.TileHeightInPixels);

        if (map != null)
            map.AddAnimatedTileset(animatedSet);

        chkPaintAnimated.Enabled = true;

        stream.Close();
        stream.Dispose();
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error reading file.\n" + ex.Message, "Error reading image");
        return;
    }
}

void chkPaintCollision_CheckChanged(object sender, EventArgs e)
{
    if (chkPaintCollision.Checked)
    {
        chkPaintAnimated.Checked = false;
    }
```

```
}

void chkPaintAnimated_CheckChanged(object sender, EventArgs e)
{
    if (chkPaintAnimated.Checked)
    {
        chkPaintCollision.Checked = false;
    }
}
```

The code for the constructor is pretty straight forward. All it does is set properties and wire the new event handlers. The event handlers for the paint check boxes are basically the same. It checks if that control is not checked. If it is not checked it checks it and then deselects the other check box.
The code for handling the dialog should be pretty familiar at this point. First, I create an instance of the form for creating animated tile sets and display it. If the dialog was cancelled I exit the method. Next up there is a try-catch that attempts to read the texture and the image from the data. It also creates a new animated tile set. If a map exists then I call the AddAnimatedTileset method we just added to set the tile set to that object. Finally I enable the paint animated tiles check box. If there is a failure doing any of these tasks I show a message box and exit the method.

The next step will be to enable the check boxes if there is a valid map object and for the animated tile set that an animated tile set exists. I did that in the newLayerToolStripMenuItem_Click method. Update that code to the following.

```
void newLayerToolStripMenuItem_Click(object sender, EventArgs e)
{
    using (FormNewLayer frmNewLayer = new FormNewLayer(levelData.MapWidth,
        levelData.MapHeight))
    {
        frmNewLayer.ShowDialog();

        if (frmNewLayer.OKPressed)
        {
            MapLayerData data = frmNewLayer.MapLayerData;

            if (clbLayers.Items.Contains(data.MapLayerName))
            {
                MessageBox.Show("Layer with name " + data.MapLayerName + " exists.",
                    "Existing layer");
                return;
            }

            MapLayer layer = MapLayer.FromMapLayerData(data);

            clbLayers.Items.Add(data.MapLayerName, true);
            clbLayers.SelectedIndex = clbLayers.Items.Count - 1;

            layers.Add(layer);

            if (map == null)
            {
                map = new TileMap(tileSets[0], animatedSet, (MapLayer)layers[0]);

                for (int i = 1; i < tileSets.Count; i++)
                {
                    map.AddTileset(tileSets[1]);
```

```
                }

                for (int i = 1; i < layers.Count; i++)
                {
                    map.AddLayer(layers[1]);
                }
            }

            charactersToolStripMenuItem.Enabled = true;
            chestsToolStripMenuItem.Enabled = true;
            keysToolStripMenuItem.Enabled = true;

            if (animatedSet != null)
            {
                chkPaintAnimated.Enabled = true;
            }

            chkPaintCollision.Enabled = true;
        }
    }
}
```

The next step will be to handle painting collisions and animated tiles with the editor. To do that I add a field for the collision layer, updated the Logic method and added two new methods, PaintAnimated and PaintCollision which paint an animated tile and a collision type. Update the Logic method to the following and add the new methods and field.

```
public static CollisionLayer collisionLayer = new CollisionLayer();

private void Logic()
{
    if (layers.Count == 0)
        return;

    Vector2 position = camera.Position;

    if (trackMouse)
    {
        if (frameCount == 0)
        {
            if (mouse.X < Engine.TileWidth)
            {
                position.X -= Engine.TileWidth;
            }

            if (mouse.X > mapDisplay.Width - Engine.TileWidth)
            {
                position.X += Engine.TileWidth;
            }

            if (mouse.Y < Engine.TileHeight)
            {
                position.Y -= Engine.TileHeight;
            }

            if (mouse.Y > mapDisplay.Height - Engine.TileHeight)
            {
                position.Y += Engine.TileHeight;
            }
```

```
            camera.Position = position;
            camera.LockCamera();
        }

        position.X = mouse.X + camera.Position.X;
        position.Y = mouse.Y + camera.Position.Y;

        Point tile = Engine.VectorToCell(position);

        tbMapLocation.Text =
            "( " + tile.X.ToString() + ", " + tile.Y.ToString() + " )";

        if (isMouseDown && !chkPaintAnimated.Checked && !chkPaintCollision.Checked)
        {
            if (rbDraw.Checked)
            {
                SetTiles(tile, (int)nudCurrentTile.Value, lbTileset.SelectedIndex);
            }

            if (rbErase.Checked)
            {
                SetTiles(tile, -1, -1);
            }
        }
        else if (isMouseDown && chkPaintAnimated.Checked)
        {
            PaintAnimated(tile);
        }
        else if (isMouseDown && chkPaintCollision.Checked)
        {
            CollisionType cType = CollisionType.Passable;

            if (rbImpassable.Checked)
            {
                cType = CollisionType.Impassable;
            }

            PaintCollision(tile, cType);
        }
    }
}

private void PaintAnimated(Point tile)
{
    if ((int)sbAnimatedTile.Value == -1 &&
        map.AnimatedTileLayer.AnimatedTiles.ContainsKey(tile))
    {
        map.AnimatedTileLayer.AnimatedTiles.Remove(tile);
        return;
    }

    if (map.AnimatedTileLayer.AnimatedTiles.ContainsKey(tile))
    {
        map.AnimatedTileLayer.AnimatedTiles[tile].TileIndex = (int)sbAnimatedTile.Value;
    }
    else
    {
        map.AnimatedTileLayer.AnimatedTiles.Add(tile, new
            AnimatedTile((int)sbAnimatedTile.Value, animatedSet.FrameCount));
```

```
    }
}

private void PaintCollision(Point tile, CollisionType collisionValue)
{
    if (collisionValue == CollisionType.Passable &&
        collisionLayer.Collisions.ContainsKey(tile))
    {
        collisionLayer.Collisions.Remove(tile);
        return;
    }

    if (collisionLayer.Collisions.ContainsKey(tile))
    {
        collisionLayer.Collisions[tile] = collisionValue;
    }
    else
    {
        collisionLayer.Collisions.Add(tile, collisionValue);
    }
}
```

In the logic method I added in two else if statements after I check the isMouseDown. The first one checks if isMouseDown is selected and chkPaintAnimated is checked. If those conditions are true I call the PaintAnimated passing in the tile. The second else if checks if isMouseDown is true and chkPaintCollision is checked. In that case I create a CollisionType variable and set it to be Passable. Next there is an if statement that checks if rbImpassable is checked. If it is I update the variable to be Impassable. I then call PaintCollision passing in the tile and the type of collision.

In PaintAnimated I first check to see if the selected value in the numeric up down for the animated tile is -1 and then if an animated tile exists with the tile as a key. If both are true I remove the tile and exit the method. I then check if key exists for the tile. If it does I set the TileIndex property to the value of the numeric up down. If it does not exist I add a new item to the collection.

The PaintCollision method works similarly. I check to see if the collision type is passable and if the collision collection contains that key. If it does I remove the entry and exit the method. Then if the tile exists as a key update the value or if it does not add a new collision to the collection.

That leaves rendering the animated tiles and a visual cue that a tile has a collision associated with it. First, I needed to make a change to the animatedSet field and add a field for an AnimatedTileLayer. They both need to be static so they can be accessed in the MapDisplay class. Update the fields related to animated tiles to the following.

```
public static AnimatedTileset animatedSet;
AnimatedTilesetData animatedSetData;
public static AnimatedTileLayer animatedLayer;
```

Rendering will be done in the Draw method of the MapDisplay class. Update that method as follows and add the new method PaintCollision.

```
protected override void Draw()
{
    base.Draw();
```

```
    shadowPosition = new Vector2(mouseState.Position.X, mouseState.Position.Y) +
        FormMain.camera.Position;
    Point p = Engine.VectorToCell(shadowPosition);

    for (int i = 0; i < FormMain.layers.Count; i++)
    {
        Editor.spriteBatch.Begin(
            SpriteSortMode.Deferred,
            BlendState.AlphaBlend,
            SamplerState.PointClamp,
            null,
            null,
            null,
            FormMain.camera.Transformation);

        FormMain.layers[i].Draw(Editor.spriteBatch, FormMain.camera, FormMain.tileSets);

        Rectangle destination = new Rectangle(
            (int)p.X * Engine.TileWidth,
            (int)p.Y * Engine.TileHeight,
            FormMain.brushWidth * Engine.TileWidth,
            FormMain.brushWidth * Engine.TileHeight);


        Color tint = Color.White;
        tint.A = 1;
        Editor.spriteBatch.Draw(shadow, destination, tint);
        Editor.spriteBatch.End();
    }

    Editor.spriteBatch.Begin(
        SpriteSortMode.Deferred,
        BlendState.AlphaBlend,
        SamplerState.PointClamp,
        null,
        null,
        null,
        FormMain.camera.Transformation);

    if (FormMain.animatedSet != null && FormMain.animatedLayer.AnimatedTiles.Count > 0)
        FormMain.animatedLayer.Draw(Editor.spriteBatch, FormMain.animatedSet);

    if (FormMain.collisionLayer.Collisions.Count > 0)
        DrawCollisions();

    Editor.spriteBatch.End();

    DrawDisplay();
}

private void DrawCollisions()
{
    Color lowAlpha = Color.White;

    Texture2D collisionShadow = new Texture2D(GraphicsDevice, 32, 32, false,
        SurfaceFormat.Color);

    Color[] data = new Color[collisionShadow.Width * collisionShadow.Height];
```

```
    Color tint = Color.White;

    for (int i = 0; i < collisionShadow.Width * collisionShadow.Height; i++)
        data[i] = tint;

    collisionShadow.SetData<Color>(data);

    foreach (Point p in FormMain.collisionLayer.Collisions.Keys)
    {
        Rectangle r = new Rectangle(p.X * Engine.TileWidth, p.Y * Engine.TileHeight,
            Engine.TileWidth, Engine.TileHeight);

        if (FormMain.collisionLayer.Collisions[p] == CollisionType.Impassable)
        {
            lowAlpha = Color.Red;
            lowAlpha.A = 50;
        }

        Editor.spriteBatch.Draw(collisionShadow, r, lowAlpha);
    }
}
```

The first change to the Render method was after rendering the layers was to check that the animatedSet member variable is not null, which means the user created an animated tile set, and that there are animated tiles to be drawn. If both are true I call the Draw method of the animated tile layer. There is then another if statement that checks if there have been any tile collisions added. If there were I call the new method DrawCollisions.

In the DrawCollisions method I have a local variable, lowAlpha, that will be used to draw the collision type. I then create a new Texture2D that will be used to draw the collision. I create an array of Color that will hold the color data for the texture. I set that color that will be used to White. I then fill the array with the color and then fill the texture with that array.

There is then a foreach loop where I cycle over the key collection for the collisions. Inside I create a rectangle object using the key. There is then an if statement that checks what type of collision there is. I did that in the case that you add another type of collision type. Inside the if statement I set the tint color to red and half transparent. This works because the texture was created as solid white. That means whatever color I use in the call to draw the white will become that color. I use this often when I need a texture that is the same, other than the color. I draw it in shades of white and grey and tint it with a color. This allows for reuse of the base image and less assets.

If you were to build and run now you can place animated tiles and have them show up, but not update. Also, the tiles that have a collision associated with them will be drawn in semi-transparent red color.

I'm going to update the method for saving the level. What I've done is pass in the actual animated tile layer and collision layer to the MapData constructor. That saves those new layers to disk. Update the saveLevelToolStripMenuItem_Click method to the following.

```
void saveLevelToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (map == null)
```

```
    return;

List<MapLayerData> mapLayerData = new List<MapLayerData>();

for (int i = 0; i < clbLayers.Items.Count; i++)
{
    if (layers[i] is MapLayer)
    {
        MapLayerData data = new MapLayerData(
            clbLayers.Items[i].ToString(),
            ((MapLayer)layers[i]).Width,
            ((MapLayer)layers[i]).Height);

        for (int y = 0; y < ((MapLayer)layers[i]).Height; y++)
        {
            for (int x = 0; x < ((MapLayer)layers[i]).Width; x++)
            {
                data.SetTile(
                    x,
                    y,
                    ((MapLayer)layers[i]).GetTile(x, y).TileIndex,
                    ((MapLayer)layers[i]).GetTile(x, y).Tileset);
            }
        }

        mapLayerData.Add(data);
    }
}

MapData mapData = new MapData(levelData.MapName, tileSetData, animatedSetData,
    mapLayerData, collisionLayer, animatedLayer);

FolderBrowserDialog fbDialog = new FolderBrowserDialog
{
    Description = "Select Game Folder",
    SelectedPath = Application.StartupPath
};

DialogResult result = fbDialog.ShowDialog();

if (result == DialogResult.OK)
{
    if (!File.Exists(fbDialog.SelectedPath + @"\Game.xml"))
    {
        MessageBox.Show("Game not found", "Error");
        return;
    }

    string LevelPath = Path.Combine(fbDialog.SelectedPath, @"Levels\");
    string MapPath = Path.Combine(LevelPath, @"Maps\");

    if (!Directory.Exists(LevelPath))
    {
        Directory.CreateDirectory(LevelPath);
    }

    if (!Directory.Exists(MapPath))
    {
        Directory.CreateDirectory(MapPath);
    }
```

```
        XnaSerializer.Serialize<LevelData>(LevelPath + levelData.LevelName + ".xml",
            levelData);
        XnaSerializer.Serialize<MapData>(MapPath + mapData.MapName + ".xml", mapData);
    }
}
```

There is two quick fixes that I need to make to FormMain of XlevelEditor. Firstly, the initial brush width is set to 0 and you cannot paint. Find the brushWidth field and set it to the following.

```
        public static int brushWidth = 1;
```

Second, is that the event handlers for the mouse and the map display have been removed somewhere along the line. Replace the code for the constructor of FormMain to the following.

```
public FormMain()
{
    InitializeComponent();

    this.Load += new EventHandler(FormMain_Load);
    this.FormClosing += new FormClosingEventHandler(FormMain_FormClosing);

    tilesetToolStripMenuItem.Enabled = false;
    mapLayerToolStripMenuItem.Enabled = false;
    charactersToolStripMenuItem.Enabled = false;
    chestsToolStripMenuItem.Enabled = false;
    keysToolStripMenuItem.Enabled = false;
    chkPaintAnimated.Enabled = false;
    chkPaintCollision.Enabled = false;

    newLevelToolStripMenuItem.Click += new EventHandler(newLevelToolStripMenuItem_Click);
    newTilesetToolStripMenuItem.Click += new EventHandler(newTilesetToolStripMenuItem_Click);
    newLayerToolStripMenuItem.Click += new EventHandler(newLayerToolStripMenuItem_Click);
    saveLevelToolStripMenuItem.Click += new EventHandler(saveLevelToolStripMenuItem_Click);
    openLevelToolStripMenuItem.Click += new EventHandler(openLevelToolStripMenuItem_Click);

    x1ToolStripMenuItem.Click += new EventHandler(x1ToolStripMenuItem_Click);
    x2ToolStripMenuItem.Click += new EventHandler(x2ToolStripMenuItem_Click);
    x4ToolStripMenuItem.Click += new EventHandler(x4ToolStripMenuItem_Click);
    x8ToolStripMenuItem.Click += new EventHandler(x8ToolStripMenuItem_Click);

    blackToolStripMenuItem.Click += new EventHandler(blackToolStripMenuItem_Click);
    blueToolStripMenuItem.Click += new EventHandler(blueToolStripMenuItem_Click);
    redToolStripMenuItem.Click += new EventHandler(redToolStripMenuItem_Click);
    greenToolStripMenuItem.Click += new EventHandler(greenToolStripMenuItem_Click);
    yellowToolStripMenuItem.Click += new EventHandler(yellowToolStripMenuItem_Click);
    whiteToolStripMenuItem.Click += new EventHandler(whiteToolStripMenuItem_Click);

    saveTilesetToolStripMenuItem.Click += new EventHandler(saveTilesetToolStripMenuItem_Click);
    saveLayerToolStripMenuItem.Click += new EventHandler(saveLayerToolStripMenuItem_Click);

    openTilesetToolStripMenuItem.Click += new EventHandler(openTilesetToolStripMenuItem_Click);
    openLayerToolStripMenuItem.Click += new EventHandler(openLayerToolStripMenuItem_Click);

    animatedTIlesetToolStripMenuItem.Click += new
EventHandler(animatedTIlesetToolStripMenuItem_Click);

    chkPaintAnimated.CheckedChanged += new EventHandler(chkPaintAnimated_CheckChanged);
```

```
    chkPaintCollision.CheckedChanged += new EventHandler(chkPaintCollision_CheckChanged);

    mapDisplay.MouseEnter += mapDisplay_MouseEnter;
    mapDisplay.MouseDown += mapDisplay_MouseDown;
    mapDisplay.MouseMove += mapDisplay_MouseMove;
    mapDisplay.MouseUp += mapDisplay_MouseUp;
    mapDisplay.MouseLeave += mapDisplay_MouseLeave;
}
```

Lastly, yhere is one quick fix that I need to make and that is add a private constructor to the AnimatedTile class that requires no parameters. Add this constructor to the AnimatedTile class in the AnimatedTileset file.

```
private AnimatedTile()
{
}
```

I was thinking of combining this tutorial with the next but have decided against doing so. This one is getting on the long side and I don't want to overwhelm you with too much code at once. The next tutorial will cover loading levels into the editor.

So, I'm going to wrap up the tutorial here because I'd like to try and keep the tutorials to a reasonable length so that you don't have too much to digest at once. So, I encourage you to visit my blog at, https://cynthiamcmahon.ca/blog/, for the latest news on my tutorials and other goodness.

Good luck in your Game Programming Adventures!

Cynthia