System and Unit Test Report

C# Game Engine
Sharp Slugs
12-1-18

Sprint 1

A. User Story 1 Sprint 1: As a game designer I want a main "game" class so that I can have a foundation to create my game.
B. User Story 3 Sprint 1: As a game designer I want a graphics library so that I can have external assets appear on screen.

Scenario One:

1. Download SharpSlugsEngine.dll
2. Use the abstract functions that allow me to draw and update the game
   (a) Use the Game classes abstract classes to override Draw and Update
   (b) Use the Game's Graphics Manager to draw a Rectangle onto the screen

```
using System.Drawing;
using SharpSlugsEngine;

namespace SharpSlugsVeryTestGame
{
    class Program
    {
        static void Main(string[] args)
        {
            TestGame test = new TestGame();
            test.Run();
        }
    }

    public class TestGame : Game
    {
        protected override void Draw(GameTime gameTime)
        {
            Graphics.DrawRectangle(0, 0, 50, 20, Color.Aqua);
        }

        protected override void Update(GameTime gameTime)
        {

        }
    }
}
```

3. User should be see a rectangle appear on a Window on their screen.

Sprint 2

A. User Story 1 Sprint 2: As a game designer I want a content manager so that I can import assets.
B. User Story 2 Sprint 2: As a player I want to be able to view assets on-screen.

Scenario Two:

1. Download SharpSlugsEngine.dll
2. Use the abstract functions that allow me to draw and update the game. As well as initialize any content.
   (a) Use the Game classes abstract classes to override Draw and Update
   (b) Use the Game classes abstract initialize and content load functions to allow me to load external assets
   (c) Import an image into the Game using the Content Manager
   (d) Display the image as a Sprite by pulling it from the Content Manager

```
using SharpSlugsEngine;

namespace SharpSlugsVeryTestGame
{
    class Program
    {
        static void Main(string[] args)
        {
            TestGame test = new TestGame();
            test.Run();
        }
    }

    public class TestGame : Game
    {
        protected override void LoadContent()
        {
            Content.AddImage("../../Content/Example.png", "Example");
        }

        protected override void Draw(GameTime gameTime)
        {
            Graphics.DrawBMP("Example", 0, 0, 100, 56.25f);
        }

        protected override void Update(GameTime gameTime)
        {

        }
    }
}
```

3. The user should now see the image that they imported onto the screen.

A. User Story 3 Sprint 2: As a player I want to be able to use my keyboard/mouse to play the game.
B. User Story 4 Sprint 2: As a player I want to be able to use my gamepad to play the game.

Scenario Three:

1. Download the hit game **The Sharpest Slug** the famous multiplayer game made with the Sharp Slugs Engine.
2. Connect a controller to the computer and play as Player 1.
3. Connect a keyboard/mouse to the computer and play as Player 2.
4. Users should be able to play the game simultaneously.

Sprint 3

A. User Story 1 Sprint 3: As a game designer I want a world space coordinate system so that a larger world can be taken care of.
B. User Story 2 Sprint 3: As game designer I want a camera system so that the viewpoint can be moved around.

Scenario Four:

1. Download the SharpSlugsEngine.dll library
2. Override the Draw and Update functions
3. Use the WorldSpace in the Graphics Manager property to set the World Space
4. Now draw a rectangle using the Graphics Manager onto both World and Screen Space

```
using System.Drawing;
using SharpSlugsEngine;

namespace SharpSlugsVeryTestGame
{
    class Program
    {
        static void Main(string[] args)
        {
            TestGame test = new TestGame();
            test.Run();
        }
    }

    public class TestGame : Game
    {
        protected override void Initialize()
        {
            Graphics.SetWorldScale(new Vector2(1.28f, .72f));
        }

        protected override void Draw(GameTime gameTime)
        {
            Graphics.DrawRectangle(0, 0, 0.64f, .36f, Color.Linen);
            Graphics.DrawRectangle(640, 360, 64, 36, Color.Wheat, true, 0, 0, 0, DrawType.Screen);
        }

        protected override void Update(GameTime gameTime)
        {

        }
    }
}
```

5. User should now see two differently sized rectangles on the screen

A. User Story 3 Sprint 3: As a game designer I want a way to detect collisions between game objects so that they can interact with each other.
B. User Story 5 Sprint 3: As a game designer I want systems such as gravity and velocity so that I can have more of a real feel to it.

Scenario Five:

1. Download the SharpSlugsEngine.dll library
2. Override the Draw and Update functions
3. Use the Graphics Manager to create basic shapes with colliders
4. Set up gravity on one of the shapes and allow the Controller to move the other.

```
using System.Drawing;
using SharpSlugsEngine;
using SharpSlugsEngine.Input;
using SharpSlugsEngine.Physics;

namespace SharpSlugsVeryTestGame
{
    class Program
    {
        static void Main(string[] args)
        {
            TestGame test = new TestGame();
            test.Run();
        }
    }

    public class TestGame : Game
    {
        private Collider coll;

        protected override void LoadContent()
        {
            coll = new PolygonCollider(new Vector2(0, 1), new Vector2(1, 0), new Vector2(2, 0), new Vector2(3, 1),
                new Vector2(3, 2), new Vector2(2, 3), new Vector2(1, 3), new Vector2(0, 2));

            Sprites.Add("sprite", new Rect(this, 0, 0, 10, 10, Color.BurlyWood));
            Sprites.SetGravityY("sprite", 0.01f);
            Sprites.Display("sprite", true);
        }

        protected override void Draw(GameTime gameTime)
        {
            Graphics.DrawPolygon(coll.Vertices, Color.Red);
        }

        protected override void Update(GameTime gameTime)
        {
            foreach (XboxController controller in Controllers.XboxControllers)
            {
                if (controller.LeftStick.State.Length >= 0.25f)
                {
                    Vector2 move = controller.LeftStick.State * 50 * (float)gameTime.DeltaTime.TotalSeconds;
                    coll.Position += move;
                    break;
                }
            }

            if (Sprites.GetSprite("sprite").collider.IsTouching(coll))
            {
                Sprites.SetVelocityY("sprite", 0);
            }
        }
    }
}
```

5. When the game starts one shape will fall and be able to collide with the other.

Sprint 4

A. User Story 1 Sprint 4: As a game designer I want a serialization library so that my game may be more content-driven.

Scenario Six:

1. Download the SharpSlugsEngine.dll library
2. Override the Draw and Update functions
3. Create a SaveGame class and serialize it to a file
4. Deserialize this file into a SaveGame instance later

```
public class TestGame : Game
{
    protected override void Initialize()
    {
        SaveGame save = new SaveGame()
        {
            playerName = "Sharp Slugs Test Player",
            playerLevel = 12,
            playerXP = 536,
            playerArea = MapZone.Atlantis,
            playerStats = new PlayerStats() { dex = 3, str = 7, vit = 2, wis = 0 }
        };

        byte[] serialized = SerializationUtility.Serialize(save);

        if (File.Exists("../../Saves/save1.sav"))
        {
            File.Move("../../Saves/save1.sav", "../../Saves/save1.sav.bak");
        }

        File.WriteAllBytes("../../Saves/save1.sav", serialized);

        SaveGame deserialized = SerializationUtility.Deserialize<SaveGame>(File.ReadAllBytes("../../Saves/save1.sav"));
    }

    protected override void Draw(GameTime gameTime) { }
    protected override void Update(GameTime gameTime) { }
}
public class SaveGame
{
    public string playerName;
    public int playerLevel;
    public int playerXP;
    public MapZone playerArea;
    public PlayerStats playerStats;
}
public class PlayerStats
{
    public int dex;
    public int str;
    public int vit;
    public int wis;
}
public enum MapZone
{
    BoneForest,
    AbyssalDepths,
    Atlantis
}
```

5. The deserialized instance will be identical to the previously serialized instance