# 1 – Displaying the ship

Add the following class members:

Model shipModel;

Add load logic to the LoadContent method:

protected override void LoadContent()  
{

// Create a new SpriteBatch, which can be used to draw textures.

spriteBatch = new SpriteBatch(GraphicsDevice);

shipModel = Content.Load<Model>("Models\\p2\_wedge");

}

Add draw logic to the Draw method:

protected override void Draw(GameTime gameTime)

{

graphics.GraphicsDevice.Clear(Color.CornflowerBlue);

foreach (ModelMesh mesh in shipModel.Meshes)

{

foreach (BasicEffect effect in mesh.Effects)

{

effect.Projection = Matrix.CreatePerspectiveFieldOfView(MathHelper.ToRadians(45), GraphicsDevice.DisplayMode.AspectRatio, 1f, 260000.0f);

effect.View = Matrix.CreateLookAt(new Vector3(0, 0, -5000), Vector3.Zero, Vector3.Up);

effect.EnableDefaultLighting();

}

mesh.Draw();

}

base.Draw(gameTime);

}

# 2 – Moving the ship and camera

Add ship pos class member:

Matrix shipPos;

Initialize shipPos in LoadContent:

spriteBatch = new SpriteBatch(GraphicsDevice);

shipModel = Content.Load<Model>("Models\\p2\_wedge");

**shipPos = shipModel.Root.Transform \* Matrix.CreateRotationX(MathHelper.ToRadians(90)) \* Matrix.CreateTranslation(0, -8000, 0);**

Change draw method:

effect.Projection = Matrix.CreatePerspectiveFieldOfView(MathHelper.ToRadians(45), GraphicsDevice.DisplayMode.AspectRatio, **24000.0f**, 26000.0f);

effect.View = Matrix.CreateLookAt(new Vector3(0, 0, **25000**), Vector3.Zero, Vector3.Up);

**effect.World = shipPos;**

# 3 – Moving the ship

Change update method for input:

GamePadState state = GamePad.GetState(PlayerIndex.One);

Vector2 movement = new Vector2

(state.ThumbSticks.Left.X \* 200.0f \*

(float)gameTime.ElapsedGameTime.TotalMilliseconds / 12.0f,

(state.ThumbSticks.Left.Y \* 200.0f \*

(float)gameTime.ElapsedGameTime.TotalMilliseconds / 12.0f));

shipPos \*= Matrix.CreateTranslation(movement.X, movement.Y, 0);

# 4 – Shooting

Add bulletModel member and bullets:

Model shipModel, **bulletModel;**

**Matrix shipPos;**

**List<Matrix> bullets = new List<Matrix>();**

Add logic to load bulltet model to LoadContent:

bulletModel = Content.Load<Model>("Models\\mgun\_proj");

Add update logic for the bullets to Update method:

if (state.Buttons.A == ButtonState.Pressed)

{

bullets.Add(shipPos);

}

for (int i = bullets.Count - 1; i >= 0; i--)

{

var bullet = bullets[i];

bullet \*= Matrix.CreateTranslation(0, 200, 0);

if (bullet.Translation.Y > 10000)

{

bullets.RemoveAt(i);

}

else

{

bullets[i] = bullet;

}

}

Add draw logic, refactored:

protected override void Draw(GameTime gameTime)

{

graphics.GraphicsDevice.Clear(Color.CornflowerBlue);

**foreach (var bullet in bullets)**

**{**

**DrawModel(bulletModel, bullet);**

**}**

**DrawModel(shipModel, shipPos);**

base.Draw(gameTime);

}

Do: Extract to method - DrawModel.

# 5 – Adding enemies

Add enemies class members:

Model shipModel, bulletModel, **enemyModel**;

Matrix shipPos;

List<Matrix> bullets = new List<Matrix>();

**List<Matrix> enemies = new List<Matrix>();**

**Random randomizer = new Random();**

Load enemy model in LoadContent method:

enemyModel = Content.Load<Model>("Models\\p1\_saucer");

Add enemy logic to Update method:

if (randomizer.Next(40) == 4)

{

Matrix m = enemyModel.Root.Transform

\* Matrix.CreateRotationX(MathHelper.ToRadians(90))

\* Matrix.CreateTranslation(randomizer.Next(-6000, 6000), 10000, 0);

enemies.Add(m);

}

for (int i = enemies.Count - 1; i >= 0; i--)

{

var enemy = enemies[i];

enemy \*= Matrix.CreateTranslation(0, -200, 0);

if (enemy.Translation.Y < -10000)

{

enemies.RemoveAt(i);

}

else

{

enemies[i] = enemy;

}

}

Add logic to draw enemies in Draw method:

foreach (var enemy in enemies)

{

DrawModel(enemyModel, enemy);

}

# 6 – Adjust shooting

Add class member to hold previous state:

GamePadState previousState;

Add press once logic to Update method:

if (state.Buttons.A == ButtonState.**Pressed && previousState.Buttons.A == ButtonState.Released**)

{

bullets.Add(shipPos);

}

**previousState = state;**

7 – Can die

Add collision logic to update method:

**BoundingSphere shipSphere = new BoundingSphere(shipPos.Translation, shipModel.Meshes[0].BoundingSphere.Radius);**

for (int i = enemies.Count - 1; i >= 0; i--)

{

var enemy = enemies[i];

enemy \*= Matrix.CreateTranslation(0, -200, 0);

**BoundingSphere enemySphere = new BoundingSphere(enemies[i].Translation, enemyModel.Meshes[0].BoundingSphere.Radius);**

**if (shipSphere.Intersects(enemySphere)) Exit();**

if (enemy.Translation.Y < -10000)

{

enemies.RemoveAt(i);

}

else

{

enemies[i] = enemy;

}

}

8 – Enemies can die

Change update logic:

BoundingSphere enemySphere = new BoundingSphere(enemies[i].Translation, enemyModel.Meshes[0].BoundingSphere.Radius);

if (shipSphere.Intersects(enemySphere)) Exit();

if (enemy.Translation.Y < -10000)

{

enemies.RemoveAt(i);

}

else

{

enemies[i] = enemy;

}

**foreach (var bullet in bullets)**

**{**

**BoundingSphere bulletSphere = new BoundingSphere(bullet.Translation, bulletModel.Meshes[0].BoundingSphere.Radius);**

**if (bulletSphere.Intersects(enemySphere))**

**{**

**enemies.RemoveAt(i);**

**break;**

**}**

**}**

}Mesh is a collection of points

Effect is collection of several things:

* is a descrpiontion about how to render a certen piece on the screen.
* How do we light that
* How to position that
* A reference to shaders
  + Piece of code running on the graphics hardware
  + Writen in HLSL
  + Where to put a point in what color

Project, View, Word – Descrips via matrices.

Project: Tel the effect how the rendering space lookslike.

,