**How to Rotate and Move a Camera**

**XNA Game Studio Express**

[Other Versions](javascript:;)

https://i-msdn.sec.s-msft.com/Areas/Epx/Content/Images/ImageSprite.png?v=635718260282234284

* [XNA Game Studio 4.0](https://msdn.microsoft.com/en-US/library/bb197901(v=xnagamestudio.40).aspx)
* [XNA Game Studio 3.1](https://msdn.microsoft.com/en-US/library/bb197901(v=xnagamestudio.31).aspx)
* [XNA Game Studio 3.0](https://msdn.microsoft.com/en-US/library/bb197901(v=xnagamestudio.30).aspx)
* [XNA Game Studio 2.0](https://msdn.microsoft.com/en-US/library/bb197901(v=xnagamestudio.20).aspx)

This example demonstrates how to rotate and move the camera by creating a view matrix with [CreateLookAt](https://msdn.microsoft.com/en-US/library/microsoft.xna.framework.matrix.createlookat(v=xnagamestudio.10).aspx).

The camera's position and orientation are controlled by setting a view [Matrix](https://msdn.microsoft.com/en-US/library/microsoft.xna.framework.matrix(v=xnagamestudio.10).aspx) for the camera to use. In this example, it is assumed the camera will move frequently, so the view [Matrix](https://msdn.microsoft.com/en-US/library/microsoft.xna.framework.matrix(v=xnagamestudio.10).aspx) is created and set every time [Game.Draw](https://msdn.microsoft.com/en-US/library/microsoft.xna.framework.game.draw(v=xnagamestudio.10).aspx) is called. Similarly, it is assumed that the projection [Matrix](https://msdn.microsoft.com/en-US/library/microsoft.xna.framework.matrix(v=xnagamestudio.10).aspx) may change from frame to frame for effects such as zooming. The example assumes a model has been added to the project as described in [How to: Render a Model](https://msdn.microsoft.com/en-US/library/bb203933(v=xnagamestudio.10).aspx).

**To rotate and move the camera**

1. Determine the camera's position in world coordinates.
2. Determine the reference [Vector3](https://msdn.microsoft.com/en-US/library/microsoft.xna.framework.vector3(v=xnagamestudio.10).aspx) that rotation of the camera is relative to. The direction should not change during the game and will usually be (0, 0, 1) or (0, 0, −1).

C#

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_f8973897-0a0b-40eb-ad00-450f0cb066bb');" \o "Copy to clipboard.)

// Set the direction the camera points without rotation.

Vector3 cameraReference = new Vector3( 0, 0, 10 );

1. Create a rotation [Matrix](https://msdn.microsoft.com/en-US/library/microsoft.xna.framework.matrix(v=xnagamestudio.10).aspx) representing the direction the camera is facing. A rotation [Matrix](https://msdn.microsoft.com/en-US/library/microsoft.xna.framework.matrix(v=xnagamestudio.10).aspx) representing rotation around the y-axis would be created with [CreateRotationY](https://msdn.microsoft.com/en-US/library/microsoft.xna.framework.matrix.createrotationy(v=xnagamestudio.10).aspx).

C#

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_17277c72-3ebd-4328-90c5-a889c5ad6afa');)

Matrix rotationMatrix = Matrix.CreateRotationY( avatarYaw );

1. Transform a copy of the reference [vector](https://msdn.microsoft.com/en-US/library/microsoft.xna.framework.vector3(v=xnagamestudio.10).aspx) using [Transform](https://msdn.microsoft.com/en-US/library/microsoft.xna.framework.vector3.transform(v=xnagamestudio.10).aspx) and the rotation [Matrix](https://msdn.microsoft.com/en-US/library/microsoft.xna.framework.matrix(v=xnagamestudio.10).aspx).

C#

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_fe77de50-ded6-4759-b049-b5589ce52d69');)

// Create a vector pointing the direction the camera is facing.

Vector3 transformedReference = Vector3.Transform( cameraReference, rotationMatrix );

1. Add the camera's current position to the transformed direction [vector](https://msdn.microsoft.com/en-US/library/microsoft.xna.framework.vector3(v=xnagamestudio.10).aspx). The result is the position that the camera is looking toward.

C#

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_cc5aefe4-54a1-4366-b9b5-bf08e40d9a35');)

// Calculate the position the camera is looking at.

Vector3 cameraLookat = cameraPosition + transformedReference;

1. Create a new view [Matrix](https://msdn.microsoft.com/en-US/library/microsoft.xna.framework.matrix(v=xnagamestudio.10).aspx) using [CreateLookAt](https://msdn.microsoft.com/en-US/library/microsoft.xna.framework.matrix.createlookat(v=xnagamestudio.10).aspx). Pass [CreateLookAt](https://msdn.microsoft.com/en-US/library/microsoft.xna.framework.matrix.createlookat(v=xnagamestudio.10).aspx) the camera's current position and the transformed direction vector. The third parameter of [CreateLookAt](https://msdn.microsoft.com/en-US/library/microsoft.xna.framework.matrix.createlookat(v=xnagamestudio.10).aspx) is the up direction of the camera and will usually be [Vector3.Up](https://msdn.microsoft.com/en-US/library/microsoft.xna.framework.vector3.up(v=xnagamestudio.10).aspx) (0, 1, 0). The view [Matrix](https://msdn.microsoft.com/en-US/library/microsoft.xna.framework.matrix(v=xnagamestudio.10).aspx) controls how world coordinate values are transformed to camera coordinates.

C#

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_70d62213-e501-460f-87c1-3f048005d953');)

view = Matrix.CreateLookAt( cameraPosition, cameraLookat, new Vector3( 0.0f, 1.0f, 0.0f ) );

1. Create a new projection [Matrix](https://msdn.microsoft.com/en-US/library/microsoft.xna.framework.matrix(v=xnagamestudio.10).aspx) with [CreatePerspectiveFieldOfView](https://msdn.microsoft.com/en-US/library/microsoft.xna.framework.matrix.createperspectivefieldofview(v=xnagamestudio.10).aspx). The projection [Matrix](https://msdn.microsoft.com/en-US/library/microsoft.xna.framework.matrix(v=xnagamestudio.10).aspx) controls how camera coordinate values are transformed to screen coordinates. The first parameter is the field of view of the projection [Matrix](https://msdn.microsoft.com/en-US/library/microsoft.xna.framework.matrix(v=xnagamestudio.10).aspx) expressed in radians. A typical field of view of 45 degrees would be expressed as pi/4 radians. The second parameter is the aspect ratio of the projection [Matrix](https://msdn.microsoft.com/en-US/library/microsoft.xna.framework.matrix(v=xnagamestudio.10).aspx) and corrects for the difference in width and height of a viewspace. The third and fourth parameters specify the near and far distances that objects will be visible in.

C#

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_6e61bb47-f63a-4435-b82e-de17b538d68a');)

// Set field of view of the camera in radians (pi/4 is 45 degrees).

static float viewAngle = MathHelper.PiOver4;

// Set distance from the camera of the near and far clipping planes.

//static float nearClip = 5.0f;

static float nearClip = 1.0f;

static float farClip = 2000.0f;

Viewport viewport = graphics.GraphicsDevice.Viewport;

float aspectRatio = (float)viewport.Width / (float)viewport.Height;

proj = Matrix.CreatePerspectiveFieldOfView( viewAngle, aspectRatio, nearClip, farClip );

1. Loop through each model in the world drawing it as described in [How to: Render a Model](https://msdn.microsoft.com/en-US/library/bb203933(v=xnagamestudio.10).aspx) using the projection matrix and view matrix created above. For the world matrix, use [Matrix.CreateTranslation](https://msdn.microsoft.com/en-US/library/microsoft.xna.framework.matrix.createtranslation(v=xnagamestudio.10).aspx) and the object's current position in the world.

C#

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_885909d5-dabc-4f5c-b7af-8c72c2cec940');)

void DrawBoxes()

{

for (int z = 0; z < 9; z++)

{

for (int x = 0; x < 9; x++)

{

DrawModel( box, Matrix.CreateTranslation( x \* 60, 0, z \* 60 ), boxTexture );

}

}

}

void DrawModel( Model model, Matrix world, Texture2D texture )

{

foreach (ModelMesh mesh in model.Meshes)

{

foreach (BasicEffect be in mesh.Effects)

{

be.Projection = proj;

be.View = view;

be.World = world;

be.Texture = texture;

be.TextureEnabled = true;

}

mesh.Draw();

}

}

C#

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_b6268f72-0714-4079-8f61-482754cf0773');)

//Game1.cs

using System;

using System.Collections.Generic;

using Microsoft.Xna.Framework;

using Microsoft.Xna.Framework.Audio;

using Microsoft.Xna.Framework.Graphics;

using Microsoft.Xna.Framework.Input;

using Microsoft.Xna.Framework.Storage;

using Microsoft.Xna.Framework.Content;

class Game1 : Microsoft.Xna.Framework.Game

{

ContentManager content;

GraphicsDeviceManager graphics;

Matrix view;

Matrix proj;

Model box;

Texture2D boxTexture;

Texture2D avatarTexture;

// Set the avatar position and rotation variables.

Vector3 avatarPosition = new Vector3( 0, 0, -20 );

Vector3 avatarHeadOffset = new Vector3( 0, 10, 0 );

float avatarYaw;

// Set the direction the camera points without rotation.

Vector3 cameraReference = new Vector3( 0, 0, 10 );

Vector3 thirdPersonReference = new Vector3( 0, 500, -500 );

// Set rates in world units per 1/60th second (the default fixed-step interval).

float rotationSpeed = 1f / 60f;

//float forwardSpeed = 500f / 60f;

float forwardSpeed = 50f / 60f;

// Set field of view of the camera in radians (pi/4 is 45 degrees).

static float viewAngle = MathHelper.PiOver4;

// Set distance from the camera of the near and far clipping planes.

//static float nearClip = 5.0f;

static float nearClip = 1.0f;

static float farClip = 2000.0f;

// Set the camera state, avatar's center, first-person, third-person.

int cameraState;

bool cameraStateKeyDown;

public Game1()

{

graphics = new GraphicsDeviceManager( this );

content = new ContentManager( Services );

}

protected override void Initialize()

{

// TODO: Add your initialization logic here.

base.Initialize();

}

protected override void LoadGraphicsContent( bool loadAllContent )

{

if (loadAllContent)

{

box = content.Load<Model>( "box" );

boxTexture = content.Load<Texture2D>( "boxtexture" );

avatarTexture = content.Load<Texture2D>( "avatartexture" );

}

}

protected override void UnloadGraphicsContent( bool unloadAllContent )

{

if (unloadAllContent)

{

content.Unload();

}

}

protected override void Update( GameTime gameTime )

{

// Allows the default game to exit on Xbox 360 and Windows.

if (GamePad.GetState( PlayerIndex.One ).Buttons.Back == ButtonState.Pressed)

this.Exit();

GetCurrentCamera();

UpdateAvatarPosition();

base.Update( gameTime );

}

protected override void Draw( GameTime gameTime )

{

graphics.GraphicsDevice.Clear( Color.CornflowerBlue );

switch (cameraState)

{

default:

case 0:

UpdateCamera();

break;

case 1:

UpdateCameraFirstPerson();

break;

case 2:

UpdateCameraThirdPerson();

break;

}

//graphics.GraphicsDevice.RenderState.CullMode = CullMode.None;

DrawBoxes();

Matrix World = Matrix.CreateRotationY( avatarYaw ) \* Matrix.CreateTranslation( avatarPosition );

if (cameraState == 2)

{

DrawModel( box, World, avatarTexture );

}

base.Draw( gameTime );

}

// Update the position and direction of the avatar.

void UpdateAvatarPosition()

{

KeyboardState keyboardState = Keyboard.GetState();

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

if (keyboardState.IsKeyDown( Keys.Left ) || (currentState.DPad.Left == ButtonState.Pressed))

{

// Rotate left.

avatarYaw += rotationSpeed;

}

if (keyboardState.IsKeyDown( Keys.Right ) || (currentState.DPad.Right == ButtonState.Pressed))

{

// Rotate right.

avatarYaw -= rotationSpeed;

}

if (keyboardState.IsKeyDown( Keys.Up ) || (currentState.DPad.Up == ButtonState.Pressed))

{

Matrix forwardMovement = Matrix.CreateRotationY( avatarYaw );

Vector3 v = new Vector3( 0, 0, forwardSpeed );

v = Vector3.Transform( v, forwardMovement );

avatarPosition.Z += v.Z;

avatarPosition.X += v.X;

}

if (keyboardState.IsKeyDown( Keys.Down ) || (currentState.DPad.Down == ButtonState.Pressed))

{

Matrix forwardMovement = Matrix.CreateRotationY( avatarYaw );

Vector3 v = new Vector3( 0, 0, -forwardSpeed );

v = Vector3.Transform( v, forwardMovement );

avatarPosition.Z += v.Z;

avatarPosition.X += v.X;

}

}

void GetCurrentCamera()

{

KeyboardState keyboardState = Keyboard.GetState();

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

// Toggle the state of the camera.

if (keyboardState.IsKeyDown( Keys.Tab ) || (currentState.Buttons.LeftShoulder == ButtonState.Pressed))

{

cameraStateKeyDown = true;

}

else if (cameraStateKeyDown == true)

{

cameraStateKeyDown = false;

cameraState += 1;

cameraState %= 3;

}

}

void UpdateCamera()

{

// Calculate the camera's current position.

Vector3 cameraPosition = avatarPosition;

Matrix rotationMatrix = Matrix.CreateRotationY( avatarYaw );

// Create a vector pointing the direction the camera is facing.

Vector3 transformedReference = Vector3.Transform( cameraReference, rotationMatrix );

// Calculate the position the camera is looking at.

Vector3 cameraLookat = cameraPosition + transformedReference;

// Set up the view matrix and projection matrix.

view = Matrix.CreateLookAt( cameraPosition, cameraLookat, new Vector3( 0.0f, 1.0f, 0.0f ) );

Viewport viewport = graphics.GraphicsDevice.Viewport;

float aspectRatio = (float)viewport.Width / (float)viewport.Height;

proj = Matrix.CreatePerspectiveFieldOfView( viewAngle, aspectRatio, nearClip, farClip );

}

void UpdateCameraFirstPerson()

{

Matrix rotationMatrix = Matrix.CreateRotationY( avatarYaw );

// Transform the head offset so the camera is positioned properly relative to the avatar.

Vector3 headOffset = Vector3.Transform( avatarHeadOffset, rotationMatrix );

// Calculate the camera's current position.

Vector3 cameraPosition = avatarPosition + headOffset;

// Create a vector pointing the direction the camera is facing.

Vector3 transformedReference = Vector3.Transform( cameraReference, rotationMatrix );

// Calculate the position the camera is looking at.

Vector3 cameraLookat = transformedReference + cameraPosition;

// Set up the view matrix and projection matrix.

view = Matrix.CreateLookAt( cameraPosition, cameraLookat, new Vector3( 0.0f, 1.0f, 0.0f ) );

Viewport viewport = graphics.GraphicsDevice.Viewport;

float aspectRatio = (float)viewport.Width / (float)viewport.Height;

proj = Matrix.CreatePerspectiveFieldOfView( viewAngle, aspectRatio, nearClip, farClip );

}

void UpdateCameraThirdPerson()

{

Matrix rotationMatrix = Matrix.CreateRotationY( avatarYaw );

// Create a vector pointing the direction the camera is facing.

Vector3 transformedReference = Vector3.Transform( thirdPersonReference, rotationMatrix );

// Calculate the position the camera is looking from.

Vector3 cameraPosition = transformedReference + avatarPosition;

// Set up the view matrix and projection matrix.

view = Matrix.CreateLookAt( cameraPosition, avatarPosition, new Vector3( 0.0f, 1.0f, 0.0f ) );

Viewport viewport = graphics.GraphicsDevice.Viewport;

float aspectRatio = (float)viewport.Width / (float)viewport.Height;

proj = Matrix.CreatePerspectiveFieldOfView( viewAngle, aspectRatio, nearClip, farClip );

}

void DrawBoxes()

{

for (int z = 0; z < 9; z++)

{

for (int x = 0; x < 9; x++)

{

DrawModel( box, Matrix.CreateTranslation( x \* 60, 0, z \* 60 ), boxTexture );

}

}

}

void DrawModel( Model model, Matrix world, Texture2D texture )

{

foreach (ModelMesh mesh in model.Meshes)

{

foreach (BasicEffect be in mesh.Effects)

{

be.Projection = proj;

be.View = view;

be.World = world;

be.Texture = texture;

be.TextureEnabled = true;

}

mesh.Draw();

}

}

}

**How to: Make a First-Person Camera**

**XNA Game Studio Express**

[Other Versions](javascript:;)

https://i-msdn.sec.s-msft.com/Areas/Epx/Content/Images/ImageSprite.png?v=635718260282234284

* [XNA Game Studio 3.1](https://msdn.microsoft.com/en-us/library/bb203907(v=xnagamestudio.31).aspx)
* [XNA Game Studio 3.0](https://msdn.microsoft.com/en-us/library/bb203907(v=xnagamestudio.30).aspx)
* [XNA Game Studio 2.0](https://msdn.microsoft.com/en-us/library/bb203907(v=xnagamestudio.20).aspx)

This example demonstrates how to create a first-person camera.

The example controls the camera by using the method shown in [How to: Rotate and Move a Camera](https://msdn.microsoft.com/en-us/library/bb197901(v=xnagamestudio.10).aspx), with an additional offset for the camera to place it where the game avatar's head should be. The example assumes a model has been added to the project as described in [How to: Render a Model](https://msdn.microsoft.com/en-us/library/bb203933(v=xnagamestudio.10).aspx).

**To make a first-person camera**

1. Precalculate the camera's offset from the avatar. The offset is used to place the camera roughly where the avatar's head is.

C#

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_f240a07e-c7a0-41b3-9cdd-5f50d2abf569');)

Vector3 avatarHeadOffset = new Vector3( 0, 10, 0 );

1. Track the position and rotation of the avatar during gameplay.
2. Create a rotation [Matrix](https://msdn.microsoft.com/en-us/library/microsoft.xna.framework.matrix(v=xnagamestudio.10).aspx) using [CreateRotationY](https://msdn.microsoft.com/en-us/library/microsoft.xna.framework.matrix.createrotationy(v=xnagamestudio.10).aspx) and the avatar's current direction.

C#

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_196c8095-51ac-491b-9e7d-a828a87f0c81');)

Matrix rotationMatrix = Matrix.CreateRotationY( avatarYaw );

1. Transform a copy of the camera's head offset using the rotation [Matrix](https://msdn.microsoft.com/en-us/library/microsoft.xna.framework.matrix(v=xnagamestudio.10).aspx) and [Vector3.Transform](https://msdn.microsoft.com/en-us/library/microsoft.xna.framework.vector3.transform(v=xnagamestudio.10).aspx).

C#

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_b9fd3ef4-21c4-4675-b02e-c78de6ff005a');)

// Transform the head offset so the camera is positioned properly relative to the avatar.

Vector3 headOffset = Vector3.Transform( avatarHeadOffset, rotationMatrix );

1. Calculate the current position of the camera. The position of the camera will be the position of the avatar plus the avatar's transformed head offset.

C#

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_403c7c1e-b10b-458f-9b5e-0a0ae9b53194');)

// Calculate the camera's current position.

Vector3 cameraPosition = avatarPosition + headOffset;

1. Transform a copy of the camera's reference [vector](https://msdn.microsoft.com/en-us/library/microsoft.xna.framework.vector3(v=xnagamestudio.10).aspx) using the rotation [Matrix](https://msdn.microsoft.com/en-us/library/microsoft.xna.framework.matrix(v=xnagamestudio.10).aspx) and [Vector3.Transform](https://msdn.microsoft.com/en-us/library/microsoft.xna.framework.vector3.transform(v=xnagamestudio.10).aspx).

C#

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_b78a193d-502e-42c3-b2ab-aa31032a54ca');)

// Create a vector pointing the direction the camera is facing.

Vector3 transformedReference = Vector3.Transform( cameraReference, rotationMatrix );

1. Calculate the position the camera is looking at. This "look-at" position will be the camera's position plus the camera's transformed reference [vector](https://msdn.microsoft.com/en-us/library/microsoft.xna.framework.vector3(v=xnagamestudio.10).aspx).

C#

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_db4688f5-614f-46d3-a161-9d7c42684fe1');)

// Calculate the position the camera is looking at.

Vector3 cameraLookat = transformedReference + cameraPosition;

1. Create a new view [Matrix](https://msdn.microsoft.com/en-us/library/microsoft.xna.framework.matrix(v=xnagamestudio.10).aspx). The new [Matrix](https://msdn.microsoft.com/en-us/library/microsoft.xna.framework.matrix(v=xnagamestudio.10).aspx) is created by passing the camera position and camera lookat to [CreateLookAt](https://msdn.microsoft.com/en-us/library/microsoft.xna.framework.matrix.createlookat(v=xnagamestudio.10).aspx). The view [Matrix](https://msdn.microsoft.com/en-us/library/microsoft.xna.framework.matrix(v=xnagamestudio.10).aspx) controls how world coordinate values are transformed to camera coordinates.

C#

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_2aec836b-1afb-44a9-aa91-ec42ebef7fdf');)

view = Matrix.CreateLookAt( cameraPosition, cameraLookat, new Vector3( 0.0f, 1.0f, 0.0f ) );

1. Create a new projection [Matrix](https://msdn.microsoft.com/en-us/library/microsoft.xna.framework.matrix(v=xnagamestudio.10).aspx) with [CreatePerspectiveFieldOfView](https://msdn.microsoft.com/en-us/library/microsoft.xna.framework.matrix.createperspectivefieldofview(v=xnagamestudio.10).aspx). The projection [Matrix](https://msdn.microsoft.com/en-us/library/microsoft.xna.framework.matrix(v=xnagamestudio.10).aspx) controls how camera coordinate values are transformed to screen coordinates.

C#

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_3e98c0a9-4bde-4b87-8682-68e97529623f');)

Viewport viewport = graphics.GraphicsDevice.Viewport;

float aspectRatio = (float)viewport.Width / (float)viewport.Height;

proj = Matrix.CreatePerspectiveFieldOfView( viewAngle, aspectRatio, nearClip, farClip );

1. Loop through each model in the world drawing it as described in [How to: Render a Model](https://msdn.microsoft.com/en-us/library/bb203933(v=xnagamestudio.10).aspx) using the projection matrix and view matrix created above. For the world matrix, use [Matrix.CreateTranslation](https://msdn.microsoft.com/en-us/library/microsoft.xna.framework.matrix.createtranslation(v=xnagamestudio.10).aspx) and the object's current position in the world.

C#

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_01d05aa7-daaf-49c5-9808-da6ab73ecf54');)

void DrawBoxes()

{

for (int z = 0; z < 9; z++)

{

for (int x = 0; x < 9; x++)

{

DrawModel( box, Matrix.CreateTranslation( x \* 60, 0, z \* 60 ), boxTexture );

}

}

}

void DrawModel( Model model, Matrix world, Texture2D texture )

{

foreach (ModelMesh mesh in model.Meshes)

{

foreach (BasicEffect be in mesh.Effects)

{

be.Projection = proj;

be.View = view;

be.World = world;

be.Texture = texture;

be.TextureEnabled = true;

}

mesh.Draw();

}

}

C#

[Copy](javascript:if%20(window.epx.codeSnippet)window.epx.codeSnippet.copyCode('CodeSnippetContainerCode_34e1c3d2-fced-4d24-b8fd-bf94695b7951');)

//Game1.cs

using System;

using System.Collections.Generic;

using Microsoft.Xna.Framework;

using Microsoft.Xna.Framework.Audio;

using Microsoft.Xna.Framework.Graphics;

using Microsoft.Xna.Framework.Input;

using Microsoft.Xna.Framework.Storage;

using Microsoft.Xna.Framework.Content;

class Game1 : Microsoft.Xna.Framework.Game

{

ContentManager content;

GraphicsDeviceManager graphics;

Matrix view;

Matrix proj;

Model box;

Texture2D boxTexture;

Texture2D avatarTexture;

// Set the avatar position and rotation variables.

Vector3 avatarPosition = new Vector3( 0, 0, -20 );

Vector3 avatarHeadOffset = new Vector3( 0, 10, 0 );

float avatarYaw;

// Set the direction the camera points without rotation.

Vector3 cameraReference = new Vector3( 0, 0, 10 );

Vector3 thirdPersonReference = new Vector3( 0, 500, -500 );

// Set rates in world units per 1/60th second (the default fixed-step interval).

float rotationSpeed = 1f / 60f;

//float forwardSpeed = 500f / 60f;

float forwardSpeed = 50f / 60f;

// Set field of view of the camera in radians (pi/4 is 45 degrees).

static float viewAngle = MathHelper.PiOver4;

// Set distance from the camera of the near and far clipping planes.

//static float nearClip = 5.0f;

static float nearClip = 1.0f;

static float farClip = 2000.0f;

// Set the camera state, avatar's center, first-person, third-person.

int cameraState;

bool cameraStateKeyDown;

public Game1()

{

graphics = new GraphicsDeviceManager( this );

content = new ContentManager( Services );

}

protected override void Initialize()

{

// TODO: Add your initialization logic here.

base.Initialize();

}

protected override void LoadGraphicsContent( bool loadAllContent )

{

if (loadAllContent)

{

box = content.Load<Model>( "box" );

boxTexture = content.Load<Texture2D>( "boxtexture" );

avatarTexture = content.Load<Texture2D>( "avatartexture" );

}

}

protected override void UnloadGraphicsContent( bool unloadAllContent )

{

if (unloadAllContent)

{

content.Unload();

}

}

protected override void Update( GameTime gameTime )

{

// Allows the default game to exit on Xbox 360 and Windows.

if (GamePad.GetState( PlayerIndex.One ).Buttons.Back == ButtonState.Pressed)

this.Exit();

GetCurrentCamera();

UpdateAvatarPosition();

base.Update( gameTime );

}

protected override void Draw( GameTime gameTime )

{

graphics.GraphicsDevice.Clear( Color.CornflowerBlue );

switch (cameraState)

{

default:

case 0:

UpdateCamera();

break;

case 1:

UpdateCameraFirstPerson();

break;

case 2:

UpdateCameraThirdPerson();

break;

}

//graphics.GraphicsDevice.RenderState.CullMode = CullMode.None;

DrawBoxes();

Matrix World = Matrix.CreateRotationY( avatarYaw ) \* Matrix.CreateTranslation( avatarPosition );

if (cameraState == 2)

{

DrawModel( box, World, avatarTexture );

}

base.Draw( gameTime );

}

// Update the position and direction of the avatar.

void UpdateAvatarPosition()

{

KeyboardState keyboardState = Keyboard.GetState();

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

if (keyboardState.IsKeyDown( Keys.Left ) || (currentState.DPad.Left == ButtonState.Pressed))

{

// Rotate left.

avatarYaw += rotationSpeed;

}

if (keyboardState.IsKeyDown( Keys.Right ) || (currentState.DPad.Right == ButtonState.Pressed))

{

// Rotate right.

avatarYaw -= rotationSpeed;

}

if (keyboardState.IsKeyDown( Keys.Up ) || (currentState.DPad.Up == ButtonState.Pressed))

{

Matrix forwardMovement = Matrix.CreateRotationY( avatarYaw );

Vector3 v = new Vector3( 0, 0, forwardSpeed );

v = Vector3.Transform( v, forwardMovement );

avatarPosition.Z += v.Z;

avatarPosition.X += v.X;

}

if (keyboardState.IsKeyDown( Keys.Down ) || (currentState.DPad.Down == ButtonState.Pressed))

{

Matrix forwardMovement = Matrix.CreateRotationY( avatarYaw );

Vector3 v = new Vector3( 0, 0, -forwardSpeed );

v = Vector3.Transform( v, forwardMovement );

avatarPosition.Z += v.Z;

avatarPosition.X += v.X;

}

}

void GetCurrentCamera()

{

KeyboardState keyboardState = Keyboard.GetState();

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

// Toggle the state of the camera.

if (keyboardState.IsKeyDown( Keys.Tab ) || (currentState.Buttons.LeftShoulder == ButtonState.Pressed))

{

cameraStateKeyDown = true;

}

else if (cameraStateKeyDown == true)

{

cameraStateKeyDown = false;

cameraState += 1;

cameraState %= 3;

}

}

void UpdateCamera()

{

// Calculate the camera's current position.

Vector3 cameraPosition = avatarPosition;

Matrix rotationMatrix = Matrix.CreateRotationY( avatarYaw );

// Create a vector pointing the direction the camera is facing.

Vector3 transformedReference = Vector3.Transform( cameraReference, rotationMatrix );

// Calculate the position the camera is looking at.

Vector3 cameraLookat = cameraPosition + transformedReference;

// Set up the view matrix and projection matrix.

view = Matrix.CreateLookAt( cameraPosition, cameraLookat, new Vector3( 0.0f, 1.0f, 0.0f ) );

Viewport viewport = graphics.GraphicsDevice.Viewport;

float aspectRatio = (float)viewport.Width / (float)viewport.Height;

proj = Matrix.CreatePerspectiveFieldOfView( viewAngle, aspectRatio, nearClip, farClip );

}

void UpdateCameraFirstPerson()

{

Matrix rotationMatrix = Matrix.CreateRotationY( avatarYaw );

// Transform the head offset so the camera is positioned properly relative to the avatar.

Vector3 headOffset = Vector3.Transform( avatarHeadOffset, rotationMatrix );

// Calculate the camera's current position.

Vector3 cameraPosition = avatarPosition + headOffset;

// Create a vector pointing the direction the camera is facing.

Vector3 transformedReference = Vector3.Transform( cameraReference, rotationMatrix );

// Calculate the position the camera is looking at.

Vector3 cameraLookat = transformedReference + cameraPosition;

// Set up the view matrix and projection matrix.

view = Matrix.CreateLookAt( cameraPosition, cameraLookat, new Vector3( 0.0f, 1.0f, 0.0f ) );

Viewport viewport = graphics.GraphicsDevice.Viewport;

float aspectRatio = (float)viewport.Width / (float)viewport.Height;

proj = Matrix.CreatePerspectiveFieldOfView( viewAngle, aspectRatio, nearClip, farClip );

}

void UpdateCameraThirdPerson()

{

Matrix rotationMatrix = Matrix.CreateRotationY( avatarYaw );

// Create a vector pointing the direction the camera is facing.

Vector3 transformedReference = Vector3.Transform( thirdPersonReference, rotationMatrix );

// Calculate the position the camera is looking from.

Vector3 cameraPosition = transformedReference + avatarPosition;

// Set up the view matrix and projection matrix.

view = Matrix.CreateLookAt( cameraPosition, avatarPosition, new Vector3( 0.0f, 1.0f, 0.0f ) );

Viewport viewport = graphics.GraphicsDevice.Viewport;

float aspectRatio = (float)viewport.Width / (float)viewport.Height;

proj = Matrix.CreatePerspectiveFieldOfView( viewAngle, aspectRatio, nearClip, farClip );

}

void DrawBoxes()

{

for (int z = 0; z < 9; z++)

{

for (int x = 0; x < 9; x++)

{

DrawModel( box, Matrix.CreateTranslation( x \* 60, 0, z \* 60 ), boxTexture );

}

}

}

void DrawModel( Model model, Matrix world, Texture2D texture )

{

foreach (ModelMesh mesh in model.Meshes)

{

foreach (BasicEffect be in mesh.Effects)

{

be.Projection = proj;

be.View = view;

be.World = world;

be.Texture = texture;

be.TextureEnabled = true;

}

mesh.Draw();

}

}

}