Chapman University
CPSC 340 - Spring 2011

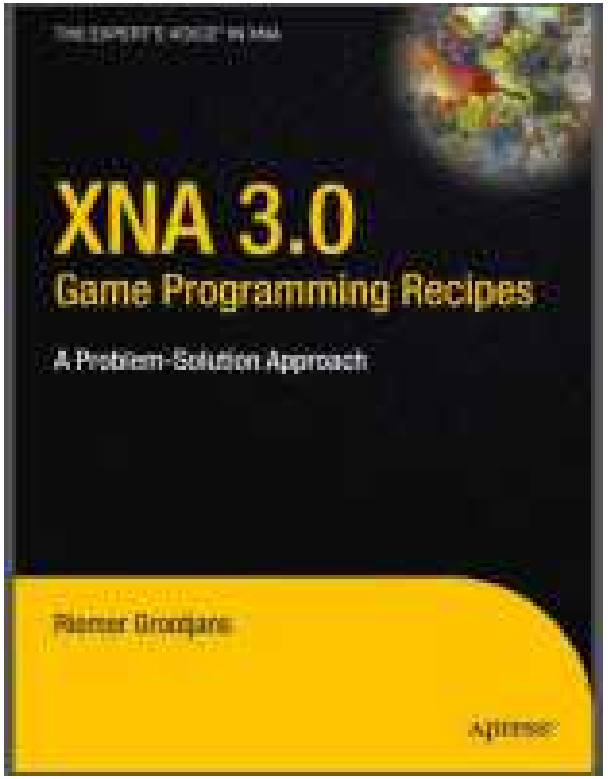
Class #5
September 29, 2011
William Wood Harter
Adjunct Professor

Summary

- Mathematical Transformations
- 3D Transformations
- 3D using XNA
- Some fun (and super useful) vector math
- Agile Software Development
- Imagine Cup

Book in PDF

- I found that the book is available for free in PDF.
- http://summerofxna.googlecode.com/files/XNA%203.0%20Game%20Programming%20Recipes.pdf

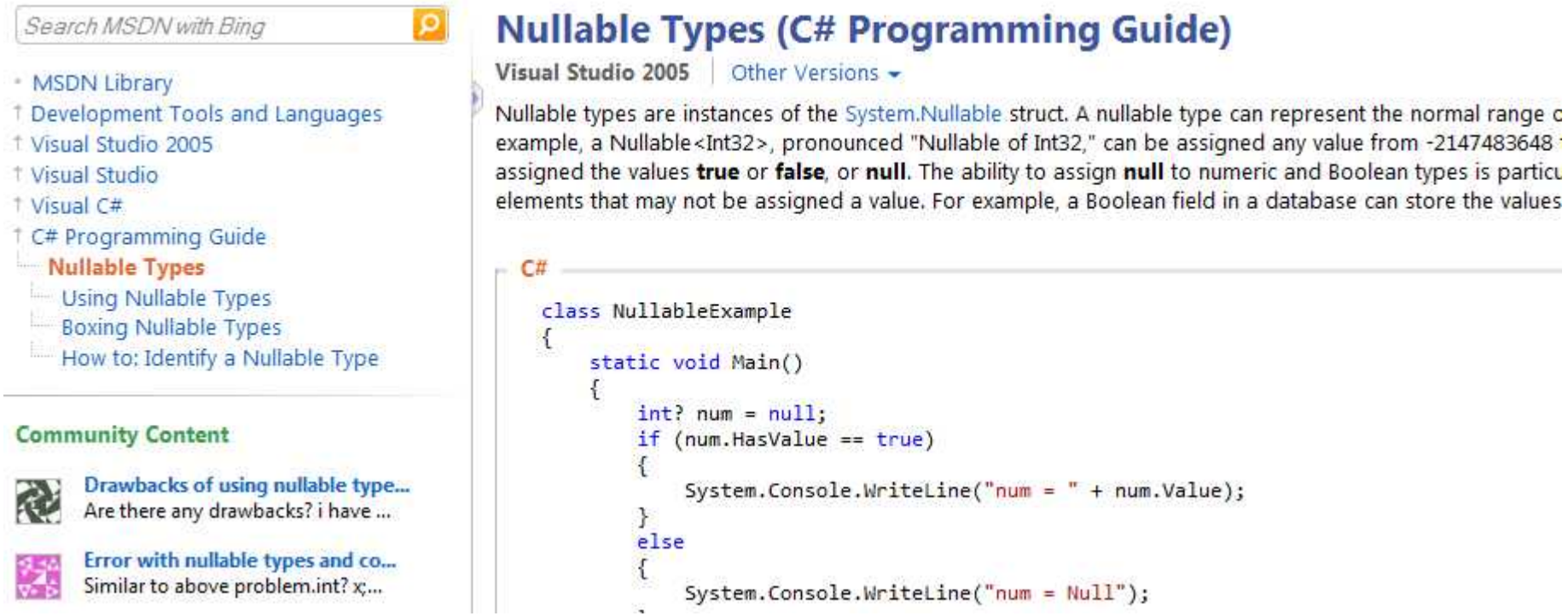


Game Timing

- There was a discussion last time that Update may be configurable as to how often it is called.
- I looked it up, this is true and outlined in the book on page 12
- By default Update is called exactly 60 times a second.
- Draw is called as often as possible
- Configurable with
  - game.TargetElapsedTime
  - game.IsFixedTimeStep
  - game.TargetElapsedTime = TimeSpan.FromSeconds(1.0f/100.0f); // 100 times/sec

C# Nullable Types

- http://msdn.microsoft.com/en-us/library/1t3y8s4s%28v=vs.80%29.aspx



Transformations

- Three basic elements to transforming a point
  - Translation
  - Scaling (maybe multiple points, ie. cube, triangle)
  - Rotation (around 0,0)

Transformations

- Translation, simply add the scaler value

$$X = X + Tx$$
$$Y = Y + Ty$$

- For scale and rotate, can use matrices
- Matrix multiplication is very useful for this.

$$\begin{bmatrix} x \\ y \end{bmatrix} \cdot \begin{bmatrix} m11 & m21 \\ m12 & m22 \end{bmatrix} = \begin{bmatrix} x * m11 + x * m21 \\ y * m12 + y * m22 \end{bmatrix}$$

  - Multiply rows times columns and sum
  - x and y are the point to multiply
  - The matrix locations have specific meaning

Transformations

- Scaling Matrix

$$\begin{bmatrix} Sx & 0 \\ 0 & Sy \end{bmatrix}$$

- Rotation Matrix

$$\begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix}$$

Combination of transforms

- Might want to combine translation, scaling and rotation in a single operation.
- Think rotation around a specific point, (translation, then rotation)
- Need to increase the size of the matrices to make them homogeneous (big math word, simple concept)

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & Tx \\ 0 & 1 & Ty \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2d coordinate, translation, scaling, rotation

Rotation in action

- Asteroids has a spaceship in the center that rotates and thrusts in a specific direction
- Pressing a button, it's easy to add or subtract some amount from a rotation angle. How do we convert that into a vector that is the direction for thrust?
- Put the angle in a rotation matrix
- Put x=0, y=-1 as the 0 angle (up)

$$\begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation in action

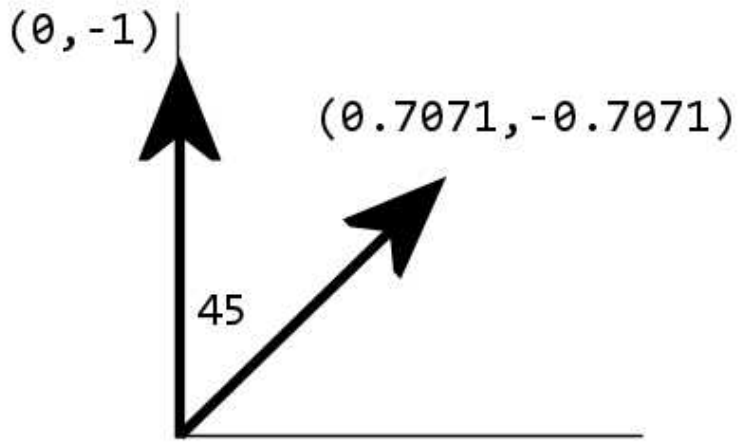
- Let's use a Theta of 45 degrees.
- Convert to radians

$$deg = rad \cdot \frac{180°}{\pi} \qquad rad = deg \cdot \frac{\pi}{180°}$$

- Most people create a 1 degree constant
  - DEGTORAD = 0.01745329251994329576923690768489
  - RADTODEG = 57.295779513082320876798154814105
- rads = degs * DEG2RAD
- Multiplications are cheaper than divisions

Rotation in Action

- Continuing with Theta of 45 degrees
- Theta = 45*RAD2DEG = 0.78539816339744830961566084581988
- Cos(theta) = 0.7071...
- Sin(theta) = 0.70710...
- Do the matrix multiplication
- X = 0*cos(theta) + -1*-sin(theta) = 0 + 0.7071 = 0.7071
- Y = 0*sin(theta) + -1*cos(theta) = 0 + -0.7071 = -0.7071



Combine Transforms

- You can combine transforms by multiplying them together.
- To rotate around a specific point, translate, then rotate.
- Create the matrices and multiply them together.
- Then just multiply all the points by the final matrix.
- Makes for very fast math
- Left as an exercise for you.

3D Transforms

- Three dimensional transforms are no different.
- Simply add z and another row and column to the matrices.
- Translation and Scale

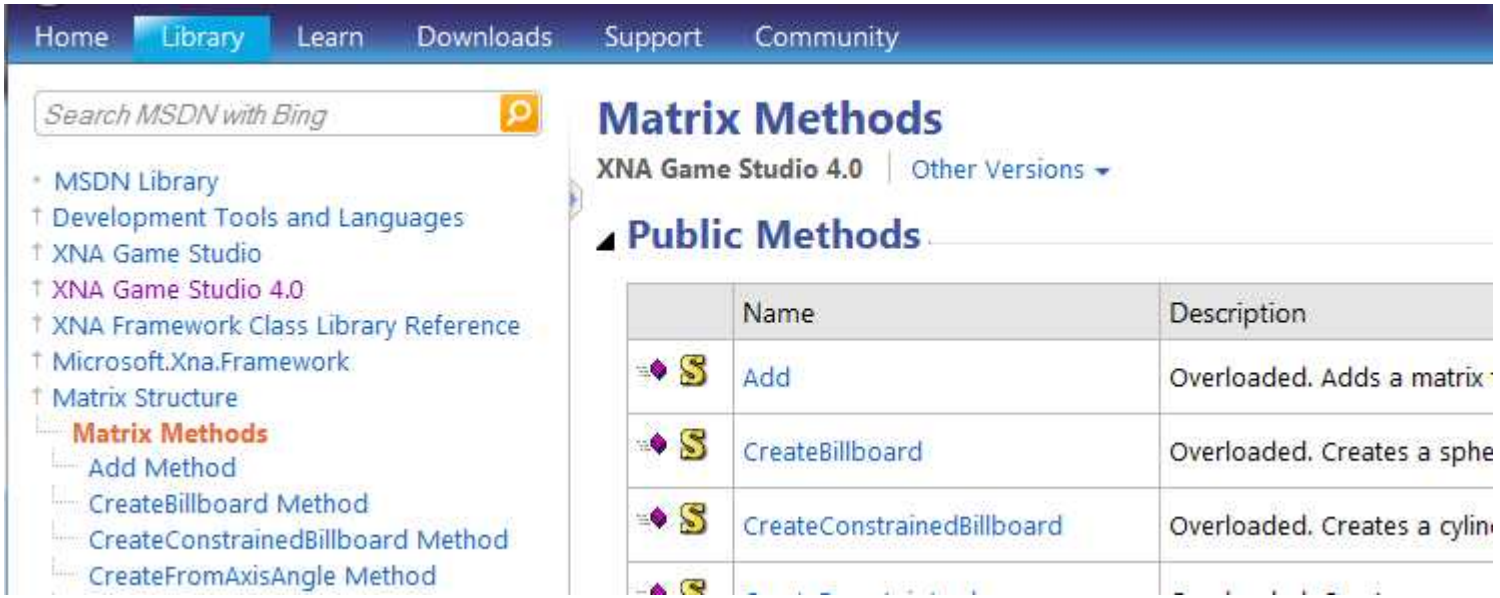
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ Tx & Ty & Tz & 1 \end{bmatrix} \quad \begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3D Transforms

- Rotations are around a specific axis

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ cos(x) & sin(x) & 0 & 0 \\ -sin(x) & cos(x) & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos(y) & -sin(y) & 0 & 0 \\ 0 & 1 & 0 & 0 \\ sin(y) & cos(y) & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos(z) & sin(z) & 0 & 0 \\ sin(z) & cos(z) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

XNA includes Matrix class



Find it on MSDN

http://msdn.microsoft.com/en-us/library/microsoft.xna.framework.matrix_members.aspx

Create different Matrices

- Matrix mTrans, mScale, mRot, mCombination;
- mTrans = Matrix.CreateTranslation(1.0f,0.0f,0.0f);
- mTrans = Matrix.CreateTranslation(myTransVect);
- mScale = Matrix.CreateScale(0.0f,1000f,0.0f);
- mRot = Matrix.CreateRotationX(rotRadians);

- mCombination = Matrix.Multiply(mTrans,mRot);

Transforming Points

- Now that you have a matrix, you can transform points with it.
- Remember my asteroids example? It goes something like this.

```
Vector3 vDir = new Vector3(0,-1, 0);
float rads = MathHelper.ToRadians(45.0f);
Matrix mRot = Matrix.CreateRotationZ(rads);
vDir = Vector3.Transform(vDir,mRot);
```

Projections

- Matrices are also used to “project” your 3d points onto a 2d plane (the screen)
- You create these matrices just like the others.
- The book describes this very well in chapter 2.

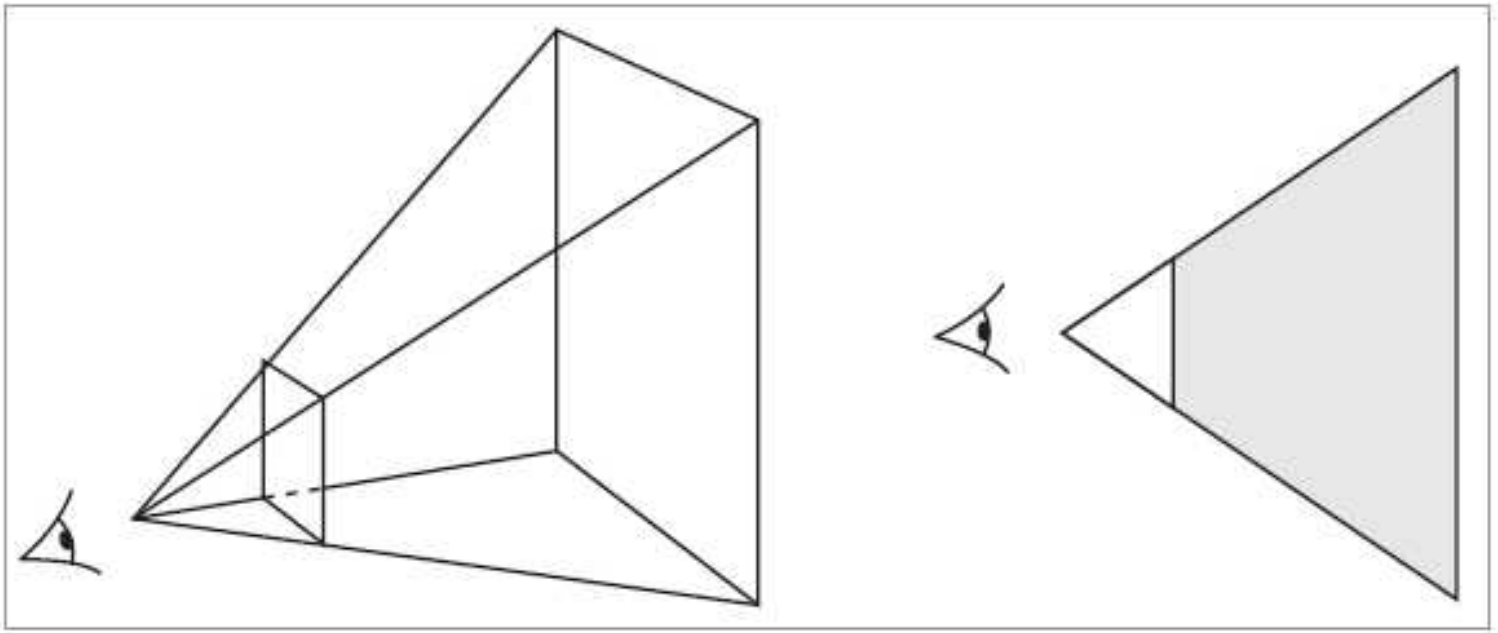


Figure 2-2. *The view frustum of a camera, 3D and cross section*

Projections

- Creating a projection matrix looks like this.

```
// create the projection matrix
float viewAngle = MathHelper.PiOver4;
aspectRatio = graphics.GraphicsDevice.Viewport.AspectRatio;
float nearPlane = 0.5f;
float farPlane = 10000.0f;
projectionMatrix = Matrix.CreatePerspectiveFieldOfView(viewAngle,
                                aspectRatio, nearPlane, farPlane);
```

View Matrix

- The projection matrix was how XNA maps to the screen.
- We also need a mapping from the camera position and orientation.
- Called the view matrix

```
// we are going to fix the camear so
// no need to recalculate every time in Update
camPosition = new Vector3(0, 0, 10.0f);
camTarget = Vector3.Zero;
camUp = Vector3.Up;
viewMatrix = Matrix.CreateLookAt(camPosition, camTarget, camUp);
```

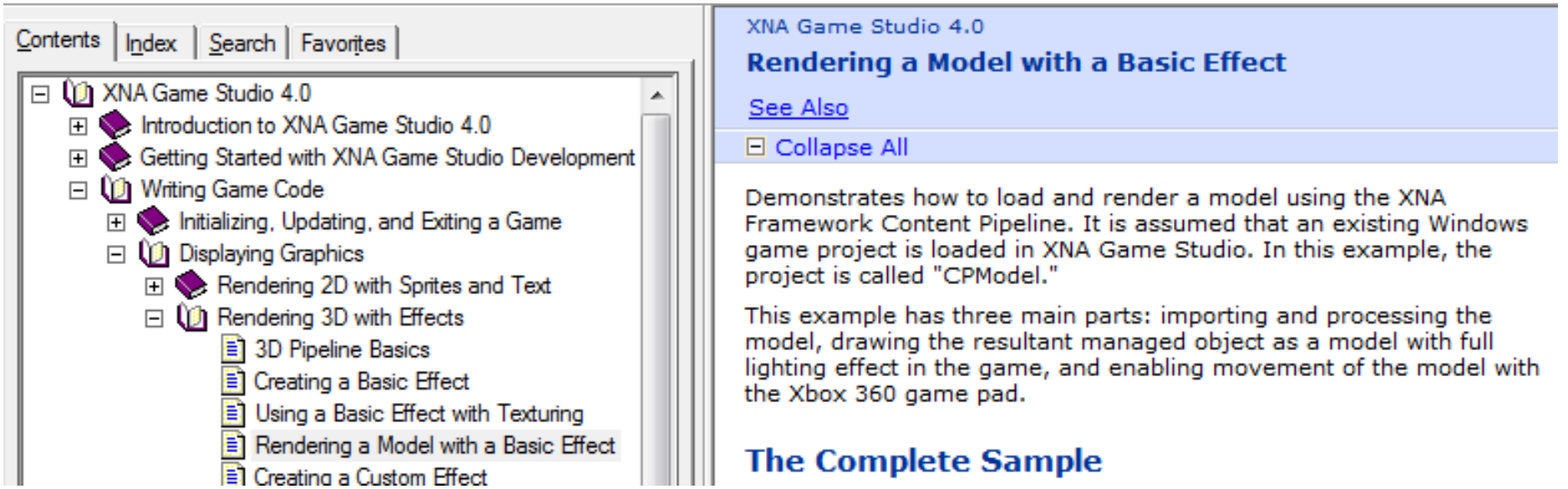
Models

- You can create models in 3dsMax
- You export them to .X or .FBX format
- Add them to the content project
- Load them just like textures.

```
protected override void LoadContent()
{
    Model modBox;
    modBox = Content.Load<Model>("box1");
```

Models

- Great example of drawing models in the XNA documentation.



DrawModel method

- I have a modified version of the DrawModel method.
- My version can scale and rotate on Y axis

```
private void DrawModel(Model m, Vector3 pos, float rotateY, float scale)
{
    // Copy any parent transforms.
    Matrix[] transforms = new Matrix[m.Bones.Count];
    m.CopyAbsoluteBoneTransformsTo(transforms);

    // Draw the model. A model can have multiple meshes, so loop.
    foreach (ModelMesh mesh in m.Meshes)
    {
        // This is where the mesh orientation is set, as well as our camera and projection.
        foreach (BasicEffect effect in mesh.Effects)
        {
            effect.EnableDefaultLighting();
            effect.World = transforms[mesh.ParentBone.Index] * Matrix.CreateRotationY(rotateY)
                * Matrix.CreateTranslation(pos) * Matrix.CreateScale(scale);
            effect.View = viewMatrix;
            effect.Projection = projectionMatrix;
        }
        // Draw the mesh, using the effects set above.
        mesh.Draw();
    }
}
```

Drawing the Model

- Once you've cut and pasted DrawModel
- Drawing any model is extremely simple
  - Parameters
    - Model
    - Position
    - Rotation on Y axis
    - Scale (why not 0.0f?)

```
if (state == STATE_PLAY)
{
    DrawModel(modBox, Vector3.Zero, MathHelper.ToRadians(rotDeg), 1.0f);
}
```

3d1

- The 3D1 project is the simplest application that draws a model.
- It is where all this code comes from.
- It is also built from my button and game state examples so it is game ready.

Dot Product

- Given two vectors you can get info about the angle between them.
- This is helpful to tell if something is in front or behind the camera. If angle is > 180 it's behind you!
- Make sure you use “Normalized” vectors!

```
v3 = Vector3.Dot( vector1, vector2 )
```

- the dot product returns a floating point value between -1 and 1 that can be used to determine some properties of the angle between two vectors. For example, it can show whether the vectors are orthogonal, parallel, or have an acute or obtuse angle between them.

v3 > 0 angle is less than 90 degrees

v3 < 0 angle is more than 90 degrees

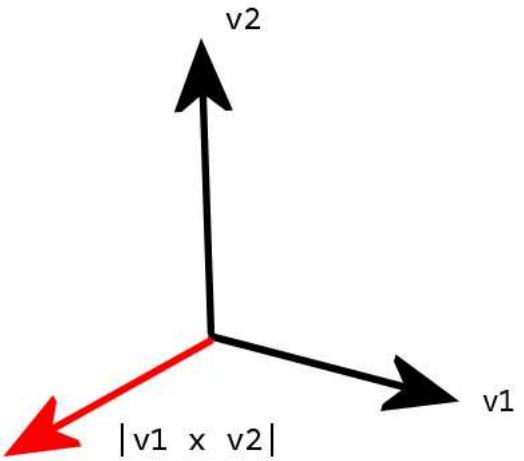
v3 == 0 angle is 90 (orthogonal)

v3 == 1 angle is 0 degress (parallel)

v3 == -1 angle is 180 (point it opposite direction)

Cross Product

- Finds the perpendicular vector given two other vectors.
- Used extensively in shaders to calculate surface normals on meshes.
- Extremely useful in finding a rotation vector to rotate two other vectors around.
- You might not know how to use it, but know it's in the tool box and build a 3D game and you'll find a use for it.
- V3 = Vector3.Cross(v1, v2);

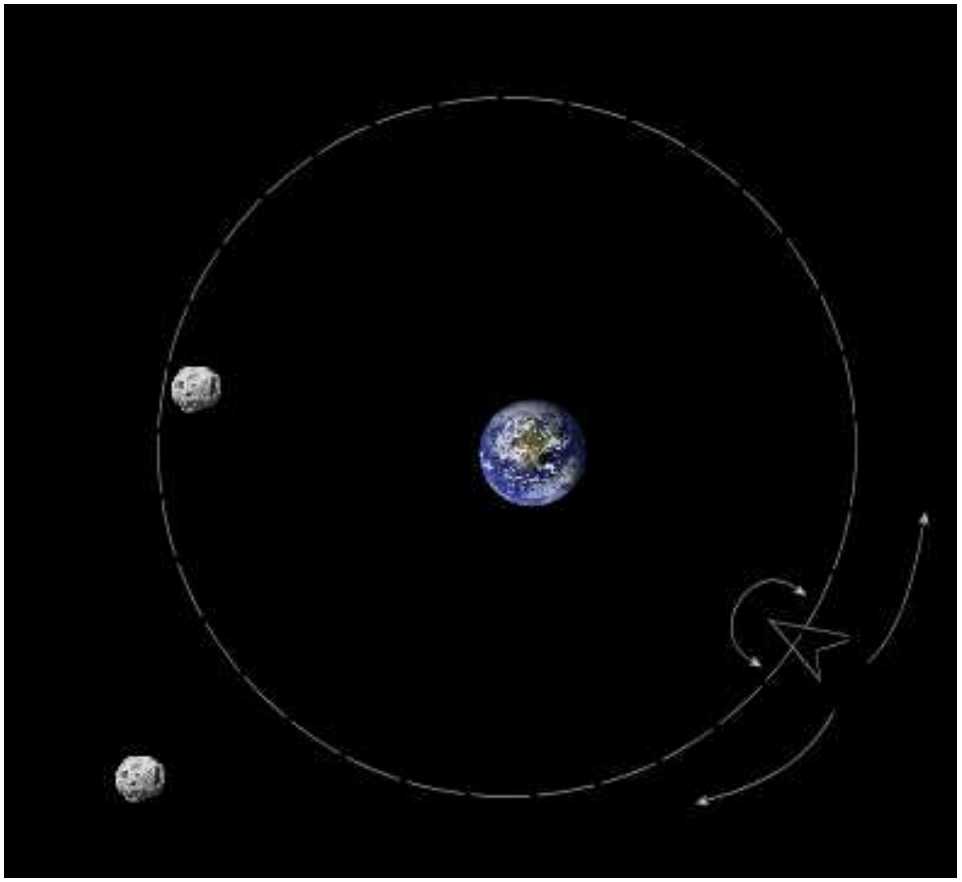


Agile Software Development

- Agile Manifesto - http://agilemanifesto.org/
  - Always have working software
  - Daily scrum (where are we today)
  - Regular planning interval (2-3 weeks)
    - Each 2-3 week iteration (sprint) finishes with working software
    - You agree to finish features during each iteration.
  - Design as you go
  - Stake holders (customers) included in process
  - Less wasted effort

Thinking In 3D

- I want to build this game, live and in class.
- A 3d version of Asteroids where the ship is locked on the XZ Plane.
- The ship can turn and move around with thrust.
- The plant attracts with gravity.
- Later, the ship rotates around the earth in a circle.
- Let's talk about the math involved.
- Next let's start with the 3d1 project we just built.



Thinking In 3D

- Create the math to rotate the camera
- Add the math to accelerate the camera
- Add gravity towards the earth
- Add a ship with the camera seeming to follow behind the ship.
- Add bullets.

Thinking in 3D

- What does the math look like if we also lock the ship in a circle around the earth?
- Are velocity and acceleration still a vector?
- What about using an angle for acceleration around the planet?

Future Lectures?

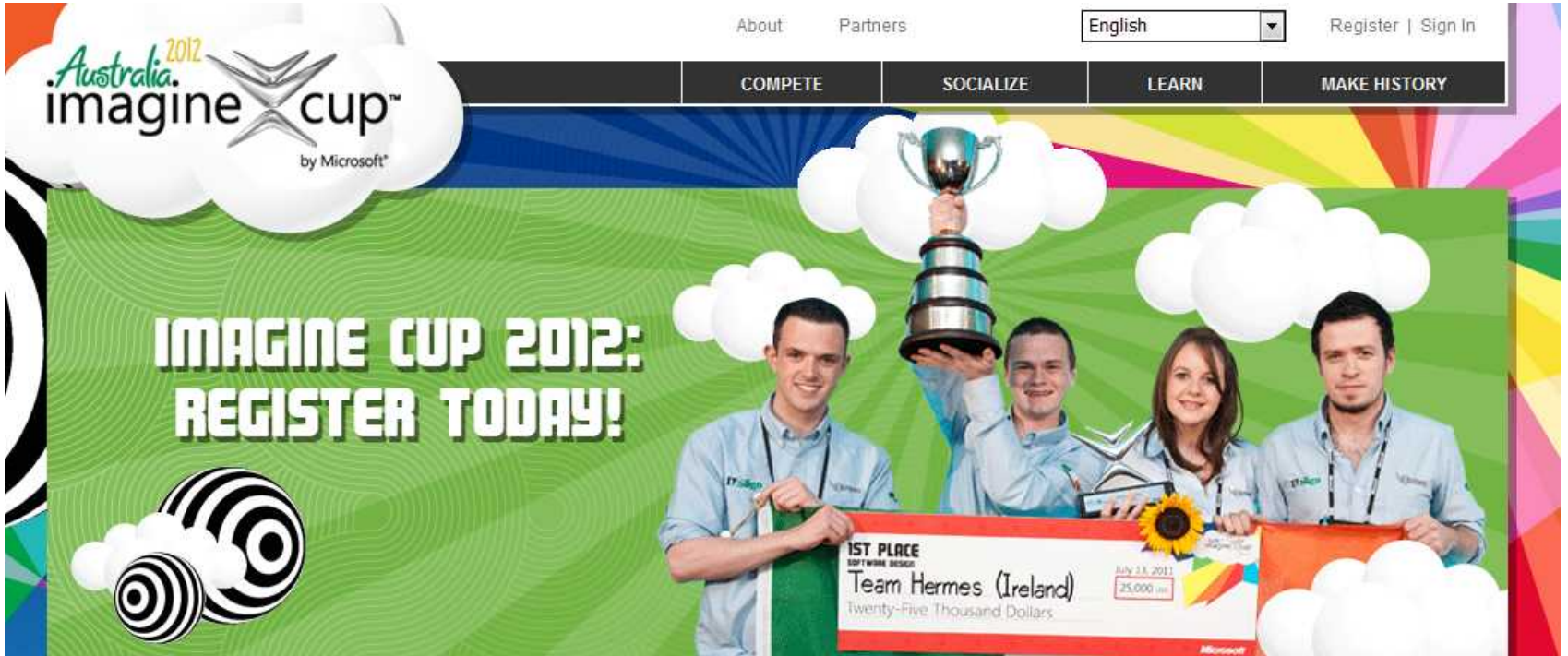
- As far as I am concerned you now have everything you need to write 3D games using XNA.
- I will lecture on your suggestions. Make suggestions here or on the mailing list.

Final projects

- Next week you will be working on your own projects
- Teams of maximum 2 (unless you convince me). Careful how you pick your team members.
- I would suggest you use a fixed camera for this game.
- Each week you will say what you will complete and I agree that is reasonable.
- Weekly project grade (100pts) will be based on your completing that functionality.
- You will simply show me the progress.

Imagine Cup

- I am willing to mentor one (and only one) team through the ImagineCup process.
- http://imaginecup.org
  - Australia 2012



Imagine Cup

- Solve the worlds toughest problems

Ending hunger and poverty

Achieving universal primary education for everyone

Promote gender equality and empower women

Reducing child mortality

Improving maternal health

Combating widespread disease

Ensuring environmental sustainability

Developing a global partnership for development

Imagine Cup

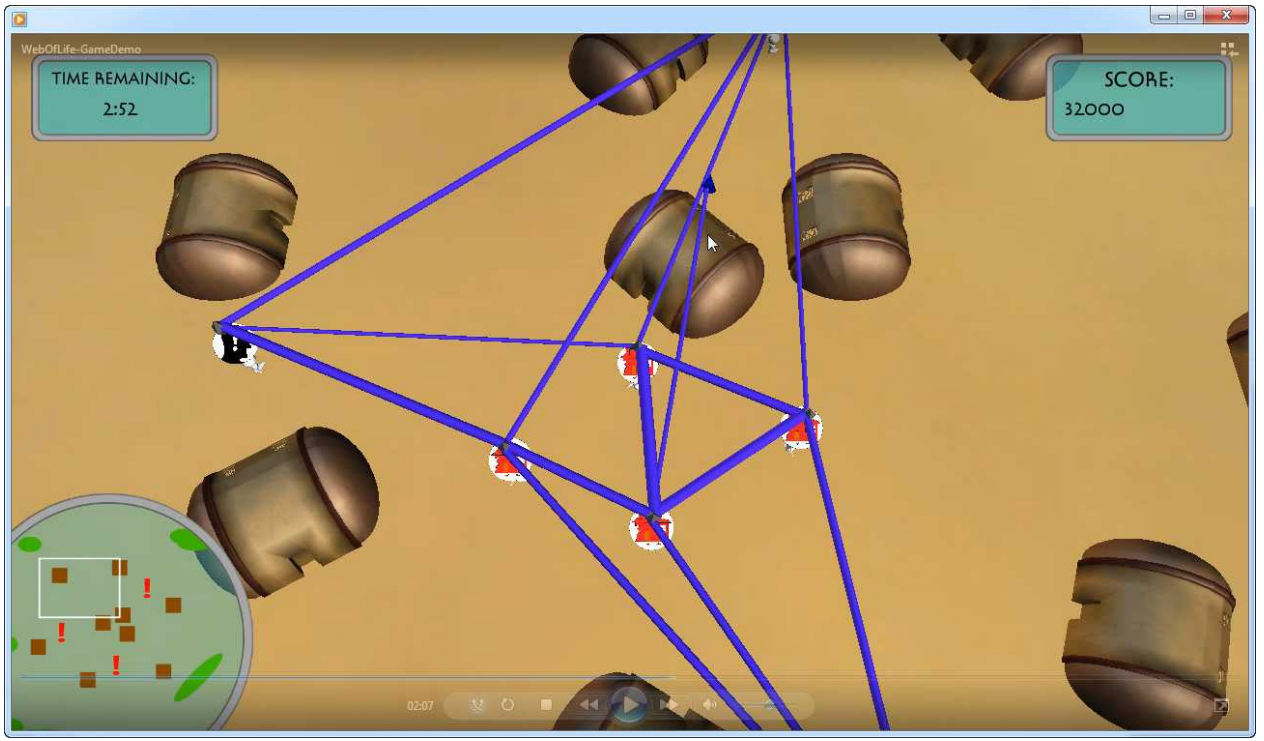
- Schedule not yet posted.
  - Usually first release due in March or April
    - Semi-finalists chosen from those submissions
  - Second release due in May/June
  - Top 5 teams picked in June.
  - July, top 5 teams go to Australia to present final version of game.

ImagineCup 2009

- 2009 - Project 2015
  - Simulate a UN Relief/Development effort
  - RTS game. Reached simi-finals 50 of 265 teams

ImagineCup 2010

- Web Of Life
  - Puzzle style game showing how connecting people in impoverished communities can improve individual situations.



Assignment #7

- Pick your team members for the final project. No approval needed for teams of less than 2. Let me know who is on your team (give it a name) or if you are going solo.
- Write a less than one page game design. And get it approved by me before next week's class. I highly suggest you send it to me before Tuesday as I may not approve it if it is too big and I don't think you can make a clean polished game by the end of the semester.
- You will design each week, so a good overview of the game would be highly recommended. No need for a giant design at the beginning.

Open Time

- Feel free to ask hard questions
- Feel free to work on your home work and ask questions while I'm still here
- Feel free to use this time for your own benefit