Updated topics schedule
Transformations
Upcoming projects

- Projects 4, 5, and 6 will be weekly team projects (teams of two)
  - Project #4 – design multiplayer protocols
  - Vote on protocols, revision contest – bonus points for team with winning protocol
  - Project #5 – create tests for protocols
  - Project #6 – Implement protocols in a game environment, apply tests

(Four weeks)
Upcoming projects

- Project #4 will involve state diagrams
  - Get the UML book

- Teams of two are **required** for project #4 – no individual work
  - Get to know your classmates
  - I will set up something for finding teammates
  - In the event that an odd number of people, I will add the last person to someone’s team
  - Individual work will be rejected
Upcoming projects

● Teammates will review each other
  – Your teammate will grade your efforts
  – These grades will be kept secret – only a semester total will be posted at the end of the class
  – Unethical conduct may result in a downward adjustment of your grade
Transformations

- From an earlier lecture:
  - Dot products can be used to convert from one coordinate system into another
  
  - Given vectors \( V, x', \) and \( y' \) expressed in \( x \) and \( y \):
    \[
    V'.x = V \cdot x' \quad \text{and} \quad V'.y = V \cdot y'
    \]
Transformations

- This can be re-expressed as follows:

\[
\begin{bmatrix}
  x'.x & x'.y \\
  y'.x & y'.y
\end{bmatrix} \cdot \begin{bmatrix}
  v.x \\
  v.y
\end{bmatrix} = \begin{bmatrix}
  v'.x \\
  v'.y
\end{bmatrix}
\]

- The elements in the rows of the first matrix are multiplied by column elements in the second to produce the third – dot products!
Transformations

- Matrix multiplication is just shorthand for this operation.

- With a n×n matrix, you can transform a point in (n-1) dimensional space as follows:
  - Translations
  - Rotations
  - Scales
  - Skews
  - Perspective effects (with additional steps)
Transformations

- It is important to remember that you are converting points via matrix multiplication (dot products) from one coordinate space into another.
  - Which is which?

- Multiple transforms can be combined in a single matrix
  - Assemble them in the proper order
Transformations in XNA

- There are two problems:
  - Drawing the sprite with a transform
  - Converting screen (or other world system) coordinates to sprite coordinates
Transformations in XNA

Drawing a transformed sprite:

Option #1: Use the draw command to anchor, rotate, flip, and scale the sprite:

```csharp
public void Draw (Texture2D texture,
     Vector2 position,
     Nullable<Rectangle> sourceRectangle,
     Color color,
     float rotation,
     Vector2 origin,
     float scale,
     SpriteEffects effects,
     float layerDepth )
```
Transformations in XNA

- Drawing a transformed sprite:
  
  **Option #2**: Create a transform matrix and apply it to the entire sprite batch:

```
Vector3.Transform(mouseLocation, Matrix.Invert(m))
Matrix m = Matrix.Identity;
  m = m * Matrix.CreateRotationZ(rotation);
  m = m * Matrix.CreateScale(2.0f);
  m = m * Matrix.CreateTranslation(300, 300, 0);

transformBatch.Begin(SpriteBlendMode.AlphaBlend, SpriteSortMode.BackToFront,
                      SaveStateMode.None, m);
```
Transformations in XNA

- The matrix $m$ converts coordinates relative to the sprite origin to coordinates on the screen.

```csharp
Matrix m = Matrix.Identity;
m = m * Matrix.CreateRotationZ(rotation);
m = m * Matrix.CreateScale(2.0f);
m = m * Matrix.CreateTranslation(300, 300, 0);
transformBatch.Begin(SpriteBlendMode.AlphaBlend, SpriteSortMode.BackToFront,
                        SaveStateMode.None, m);
```
Transformations in XNA

- The inverse matrix \( m' \) converts coordinates relative to the screen to coordinates relative to the sprite origin.

\[
\text{Matrix } m\text{\_inverse} = \text{Matrix\_Invert}(m)
\]

- The matrix can then be applied to points to convert them:

\[
\text{Vector3 } \text{spriteLoc} = \text{Matrix\_transform}(\text{mouseLoc}, m\_inverse);
\]
Transformations in XNA

- Things to remember
  - Matrices are combined by multiplying them together. The rightmost term is the first operation.
  - XNA 2D sprites support 2D transforms only.
    » The transform matrix is for 3D
    » Only 2D transforms will show up on screen
    » Tilting into z space causes strange results
  - Transform principles apply to 3D points as well, but we will not cover 3D models, lights, perspective, or cameras in class.
Transformations in XNA

- Things to remember
  - It is easier to use the modified draw command than to use a transformed sprite batch
    » But, this makes it more challenging to convert coordinates
    » You need a matrix that exactly matches the drawing transforms to convert points.
Transformations in XNA

- Classroom demos