

ECE4893A/CS4803MPG: MULTICORE AND GPU PROGRAMMING FOR VIDEO GAMES

Lecture 8 : Game Loops & XNA Content Pipeline



Prof. Aaron Lanterman

School of Electrical and Computer Engineering
Georgia Institute of Technology



Georgia Institute of Technology

Credit to where it is due

- Koen Witters
 - Thinking about game loops
- Shawn Hargreaves
 - Details about XNA's game loop
- Side note: next few slides on game loops contain rough pseudocode

Georgia Institute of Technology

Simplest game loop (1)

```
running = true;
```

```
while(running) {
    update();
    draw();
}
```

- Draw() has things like `bad_guy.x += 1;`
- What could possibly go wrong?

<http://dewitters.koonsolo.com/gameloop.html>

Georgia Institute of Technology

Simplest game loop (2)

- Game runs faster on faster hardware, slower on slower hardware
- Less of a problem if hardware is well-defined; Apple II+, Commodore 64, game console
- Try an original Mac game on a Mac II: too fast!
- Big problem on PCs/Macs with varying speed
- Can still be a problem if update time varies from iteration to iteration (i.e. varying number of bad guys)
 - See Defender and Robotron: 2084

<http://dewitters.koonsolo.com/gameloop.html>

Georgia Institute of Technology

FPS dependent on constant GS (1)

```
running = true;
seconds_per_frame = 1/60;

while(running) {
    update();
    draw();
    if (seconds_per_frame_not_elapsed_yet)
        wait(remaining_time);
    else {
        oooops! We are running behind!
    }
}
```

- What could possibly go wrong?

<http://dewitters.koonsolo.com/gameloop.html>



FPS dependent on constant GS (2)

- Slow hardware:
 - If fast enough to keep up with FPS no problem
 - If not: game will run slower
 - Worst case: some times runs normally, sometimes slower – can make unplayable
- Fast hardware:
 - Wasting cycles on desktops – higher FPS gives smoother experience, why not give that to the user?
 - Maybe not so bad philosophy on mobile devices – save battery life!
 - Also may not be bad if user is wants to run other processes

<http://dewitters.koonsolo.com/gameloop.html>



GS dependent on variable FPS (1)

```
running = true;
seconds_per_frame = 1/60;

while(running) {
    update(time_elapsed);
    draw();
}
```

- Use time_elapsed in your state update computations:

```
bad_guy.x += time_elapsed * bad_guy.velocity_x;
```

- What could possibly go wrong?

<http://dewitters.koonsolo.com/gameloop.html>



GS dependent on variable FPS (2)

- Slow hardware:
 - Game sometimes bogs down, i.e. when lots of stuff is on the screen
 - Slows down player and AI reaction time
 - If time step is too big:
 - Physics simulations may become unstable
 - "Tunneling" (need "swept collision detection")
- Fast hardware:
 - Shouldn't be a problem, right?
 - What could possibly go wrong?

<http://dewitters.koonsolo.com/gameloop.html>



GS dependent on variable FPS (3)

- Fast hardware:
 - More calculations per second for some quantity, more round off errors can accumulate
 - Multiplayer game: players with systems with different speeds will have game states drifting apart
 - Good example:
 - www.nuclex.org/articles/xna-game-loop-basics

<http://dewitters.koonsolo.com/gameloop.html>



Constant GS with max FPS (1)

```
running = true;
seconds_per_gametic = 1/50;
max_gameticks_skipped = 10;
next_gametic_time = current_time();

while (running) {
    loop = 0;
    while (current_time() > next_gametic_time
           && loops < max_gameticks_skip ) {
        update();
        loop++;
        next_gametic_time += second_per_gametic;
    }
    draw();
}
```

<http://dewitters.koonsolo.com/gameloop.html>



Constant GS with max FPS (2)

```
running = true;
seconds_per_gametic = 1/50;
max_gameticks_skipped = 10;
next_gametic_time = current_time();

while (running) {
    loop = 0;
    while (current_time() > next_gametic_time
           && loops < max_gameticks_skip ) {
        update();
        loop++;
        next_gametic_time += second_per_gametic;
    }
    draw();
}
```

- Game updated 50 times per second, render as fast as possible

- If rendering more than 50 times per second, some frames will be the same

<http://dewitters.koonsolo.com/gameloop.html>



Limits of constant GS with max FPS

- On slow hardware:
 - May have low FPS, but hopefully game will run at normal speed
 - If FPS drops below gameticks_per_second / maximum_gameticks_skipped (5 in previous example), GS slows down
- On fast hardware:
 - Wasting time redrawing the same scene (or, with better logic, twiddling thumbs)
- Balancing act: want fast update rate, but still be able to run on slow hardware

<http://dewitters.koonsolo.com/gameloop.html>



Constant GS indep. of variable FPS

- Update, at say, 25 times per second
 - Player input, AI, etc.
- Render faster on faster graphics hardware
 - Use interpolation to predict where objects should be
 - Makes it look like full game is running at a high frame rate
- Degrades gracefully on slower hardware

<http://dewitters.koonsolo.com/gameloop.html>



Tasks with different granularity

- Run often:
 - Physics engine location & orientation updates
 - 3-D character display
- Run less often:
 - Collision detection
 - Player input
 - Head-up display
- Run even less often:
 - “immediate A.I.”, networking
- Careful: A.I. might be unstable with larger time steps – not just physics!



Example: MotoGP

- Main game logic: 60 updates per second
 - “input, sound, user interface logic, camera movement, rider animations, AI, and graphical effects”
- Physics: 120 updates per second
- Networking: 4 to 30 updates per second, depending on number of players – more players results in less often updates to conserve bandwidth

<http://blogs.msdn.com/shawnhar/archive/2007/07/25/understanding-gametime.aspx>



XNA game loop: fixed step

- `Game.IsFixedTimeStep = true;` (default)
- XNA calls `Update()` “TargetElapsedTime” times per second (defaults to 60)
 - Repeat call as many times as needed to catch up with current frame (in XNA 2.0)
- XNA calls `Draw()`, then waits for next update
- If `Update+Draw time < 1/60`, we get
 - Update
 - Draw
 - Hang out for rest of time

<http://blogs.msdn.com/shawnhar/archive/2007/11/23/game-timing-in-xna-game-studio-2-0.aspx>



XNA may get behind

- Why would Update+Draw time > 1/60?
 - Computer slightly too slow
 - Computer way too slow
 - Computer mostly fast enough, but may have too much stuff on screen, big texture load garbage collection
 - Paused program in debugger
- What happens if Update+Draw time > 1/60?
 - Set `GameTime.IsRunningSlowly = true;`
 - Keep calling Update (without Draw) until caught up
 - If too far behind... punt

<http://blogs.msdn.com/shawnhar/archive/2007/07/25/understanding-gametime.aspx>



When XNA gets behind (1)

- If computer slightly too slow: If can't handle Update+Draw in one frame, can probably handle Update+Update+Draw in two frames
 - May look jerky but should play OK
- If computer way too slow (i.e. Update alone doesn't fit in a single frame): we are doomed
- In both above cases, a clever program could see that `GameTime.IsRunningSlowly == true` and reduce level of detail
 - Most games don't bother

<http://blogs.msdn.com/shawnhar/archive/2007/07/25/understanding-gametime.aspx>



When XNA gets behind (2)

- If particular frame took too long: call update extra times to catch up, then continue as normal
 - Player may notice slight glitch
- If paused in debugger: XNA will get way behind and give up, but will continue running OK when debugger resumed

<http://blogs.msdn.com/shawnhar/archive/2007/07/25/understanding-gametime.aspx>



“Heisenberg Uncertainty Principle”

- If you put in breakpoints, may notice Update being called more often than Draw, since the breakpoint makes you late
- Examining the timing of a system changes the timing!

<http://blogs.msdn.com/shawnhar/archive/2007/07/25/understanding-gametime.aspx>



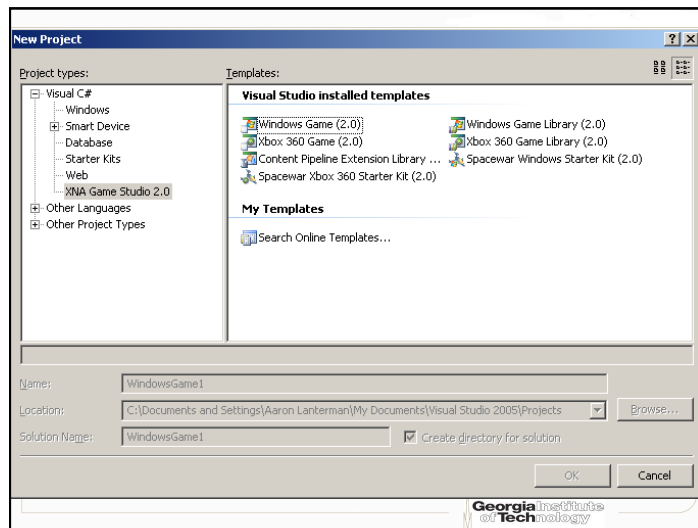
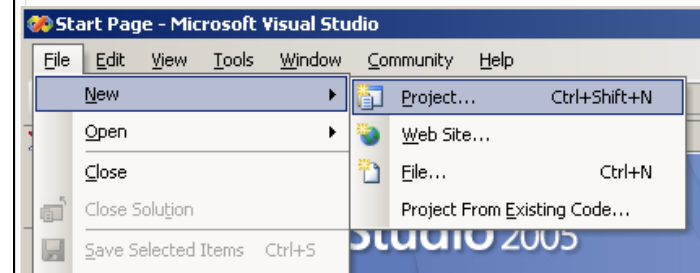
XNA game loop: Variable Step

- `Game.IsFixedTimeStep = false;`
 - Update
 - Draw
 - Repeat
 - (more or less)
- Update should use elapsed time information

<http://blogs.msdn.com/shawnhar/archive/2007/07/25/understanding-gametime.aspx>



Let's get started!



3-D file formats

- Two native formats: X (DirectX) and FBX (Autodesk) format
- Can find content pipeline importers for other formats: OBJ, MD2 (Quake 2), MD3 (Quake 3)
- Can write your own content pipeline importers

Info from "Beginning XNA 2.0 Game Programming: From Notice to Professional"



Other kinds of data

- Image (texture): BMP, DDS, DIB, HDR, JPG, PFM, PNG, and TGA
 - Can write your own content pipeline importers
- Audio: .XAP (made by XACT tool, which will import most anything you want)
- Shader: FX
- Font description: SPRITEFONT (describes how to make texture map from a specific size font)

Info from "Beginning XNA 2.0 Game Programming:
From Notice to Professional"

Georgia Institute of Technology

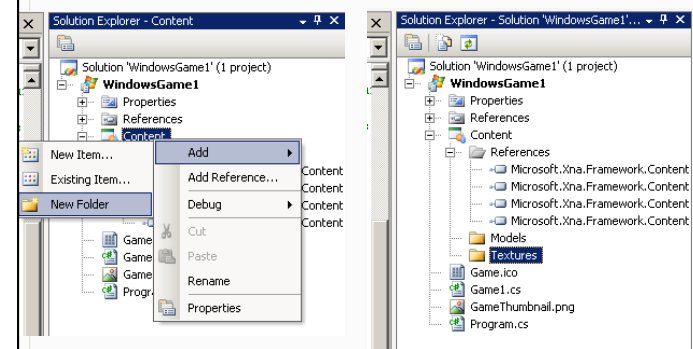


Let's find a fish!

PRODUCTS FOR DOWNLOAD						1 - 1 of 1	25	per page	1
REMOVE ALL		Name	Author	Format	Size	Expires	Price		
DOWNLOAD		red fish.rar	anima3dm	Kaydara FBX (.fbx)	320 KB	-	Free		
REMOVE									

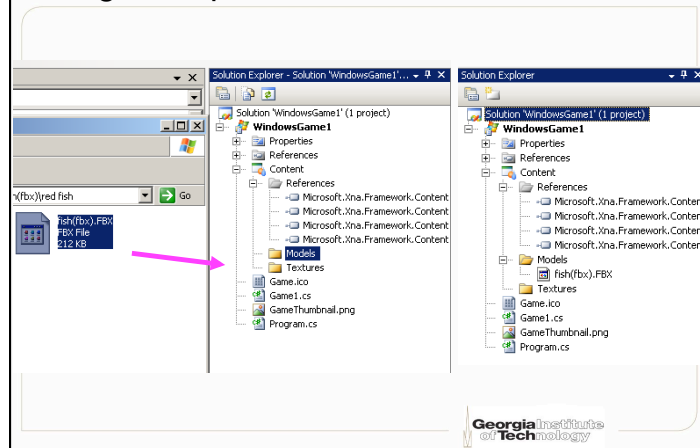
Georgia Institute of Technology

Adding some Content folders



Georgia Institute of Technology

Drag & drop a fish model



Setup for spinning fish

```
namespace WindowsGame1
{
    /// <summary>
    /// This is the main type for your game
    /// </summary>
    public class Game1 : Microsoft.Xna.Framework.Game
    {
        GraphicsDeviceManager graphics;
        SpriteBatch spriteBatch;
        Model fishModel; // Aaron added
        GraphicsDevice device; // Aaron added
        float x_rotation = 0; // Aaron added

        public Game1()
        {
            graphics = new GraphicsDeviceManager(this);
            Content.RootDirectory = "Content";
        }
    }
}
```

Load in the fish, setup effect

```
protected override void LoadContent()
{
    // Create a new SpriteBatch, which can be used to draw textures.
    spriteBatch = new SpriteBatch(GraphicsDevice);

    // TODO: use this.Content to load your game content here
    // Aaron added lines:
    fishModel = Content.Load<Model>("Models/fish(fbx)");
    device = graphics.GraphicsDevice;
    foreach (ModelMesh mesh in fishModel.Meshes)
    {
        foreach (BasicEffect effect in mesh.Effects)
        {
            float aspectRatio = (float) device.Viewport.Width /
                                device.Viewport.Height;
            effect.View = Matrix.CreateLookAt(new Vector3(100.0f, 100.0f, 0.0f),
                                              Vector3.Zero,
                                              Vector3.Up);
            effect.Projection = Matrix.CreatePerspectiveFieldOfView(
                MathHelper.ToRadians(45.0f),
                aspectRatio, 1f, 1000.0f);
            effect.EnableDefaultLighting();
        }
    }
}
```

Adapted from "Beginning XNA 2.0 Game Programming:
From Novice to Professional"

Update the fish

```
protected override void Update(GameTime gameTime)
{
    // Allows the game to exit
    if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed)
        this.Exit();

    // TODO: Add your update logic here
    x_rotation = x_rotation + 1; // Aaron added

    base.Update(gameTime);
}
```


Draw the fish

```
protected override void Draw(GameTime gameTime)
{
    graphics.GraphicsDevice.Clear(Color.CornflowerBlue);

    // TODO: Add your drawing code here
    // Aaron added
    foreach (ModelMesh mesh in fishModel.Meshes)
    {
        foreach (BasicEffect effect in mesh.Effects)
        {
            effect.World = Matrix.CreateRotationX(MathHelper.ToRadians(x_rotation)) *
                             Matrix.CreateRotationY(MathHelper.ToRadians(45.0f));
        }
        foreach (ModelMesh mesh in fishModel.Meshes) {
            mesh.Draw();
        }
    }

    base.Draw(gameTime);
}
```

Adapted from "Beginning XNA 2.0 Game Programming:
From Novice to Professional"

