

Computer Games 2012

Game Development

Dr. Mathias Lux
Klagenfurt University

libGDX features



- High-performance, cross-platform game development framework.
 - Android 2.0+
 - Windows, Linux & MacOS (Java)
- Basis for engines and games.
 - e.g. AndEngine



libGDX features



- Multiple backends
 - Jogl, LWJGL, Angle (NVIDIA 3D Vision) and the Android APIs
- Rendering through OpenGL ES
 - Low level & high level 3D



libGDX features



- High level 2D
 - Batched and cached sprite rendering
 - Bitmap fonts
 - Particle systems
 - TMX tile map rendering

libGDX features



- **Audio**

- Music and SFX from WAV, MP3 and OGG
- Fast Fourier transforms
- Audio decoding of OGG and MP3 via JNI
- Direct interface to audio device

- **File I/O**

- Abstracting Android assets, classpath and file-system



libGDX features



- **Input**

- Polling and event-based access to touch-screen/mouse and keyboard.
- Polling access to compass and accelerometer
- Vibration support
- Remote input event processing

- **Physics**

- Full JNI wrapper of box2d



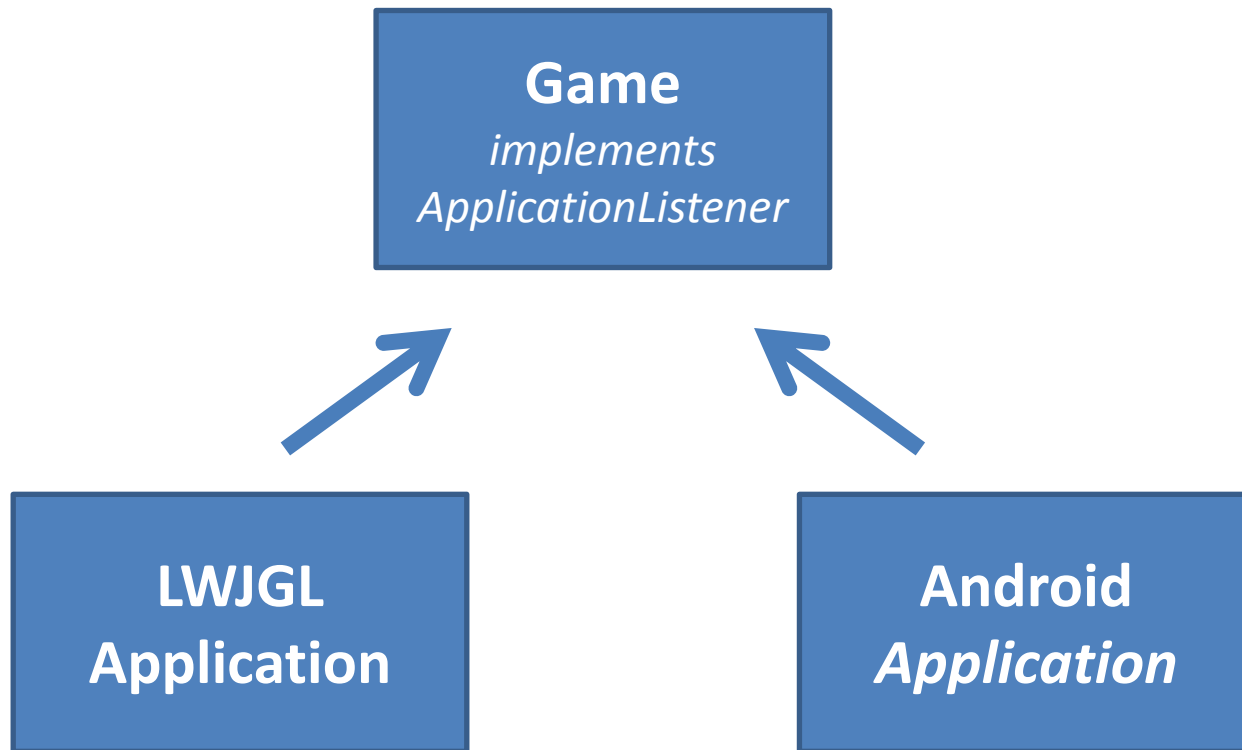
libGDX features



- Tools & Extensions
 - Particle editor
 - Hiero bitmap font generator
 - Texture packer
 - Themable Widget Library support



libGDX - Multiplattform



HelloWorld



```
public class HelloWorld implements ApplicationListener {  
    SpriteBatch spriteBatch;  
    Texture texture;  
    BitmapFont font;  
    Vector2 textPosition = new Vector2(100, 100);  
    Vector2 textDirection = new Vector2(1, 1);  
  
    @Override  
    public void create () {  
        font = new BitmapFont();  
        font.setColor(Color.RED);  
    }  
    [...]
```

HelloWorld LWJGL



```
public class HelloWorldDesktop {  
    public static void main(String[] argv) {  
        GdxTestGame game = new GdxTestGame();  
        new LwjglApplication(new HelloWorld(),  
            "Hello World", 480, 320, false);  
    }  
}
```

HelloWord Jogl



```
public class HelloWorldDesktop {  
    public static void main (String[] argv) {  
        new JoglApplication(new HelloWorld(),  
            "Hello World", 480, 320, false);  
    }  
}
```

HelloWorld Android



```
public class HelloWorldAndroid extends  
    AndroidApplication {  
    @Override  
    public void onCreate (Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        initialize(new HelloWorld(), false);  
    }  
}
```

libGdx 2D Sprites



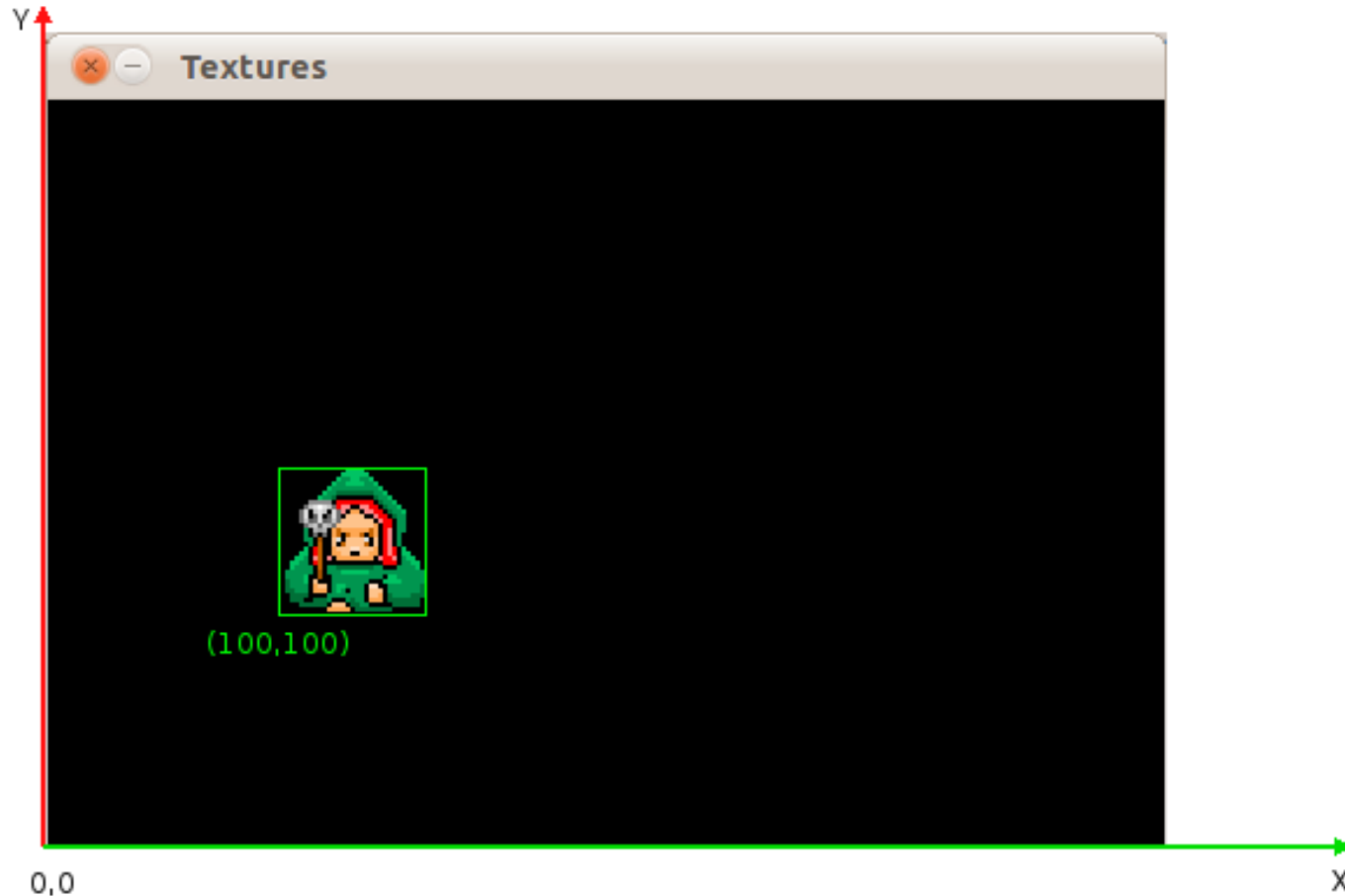
- Textures
 - OpenGL handles all sprites as Textures
 - Textures are decoded images in memory
 - Always 2^n side length (eg. 64x64 or 16x256)
- SpriteBatch
 - Takes care of displaying textures
 - Texture mapping and creation of rectangles

Drawing a single texture



```
public class TextureFun implements ApplicationListener {  
  
    private Texture druidTexture;  
    private SpriteBatch batch;  
  
    @Override  
    public void create() {  
        druidTexture = new Texture(Gdx.files.internal("druid.png"));  
        batch = new SpriteBatch();  
    }  
  
    @Override  
    public void render() {  
        batch.begin();  
        batch.draw(druidTexture, 100, 100);  
        batch.end();  
    }  
  
    // ... rest of methods omitted ... //  
}
```

Drawing a single texture



Multiple Calls



```
public void render() {  
    batch.begin();  
    batch.draw(druidTexture, 100, 100);  
    batch.draw(druidTexture, 200, 100);  
    batch.draw(druidTexture, 300, 100);  
    batch.end();  
}
```



Drawing the textures on top of each other



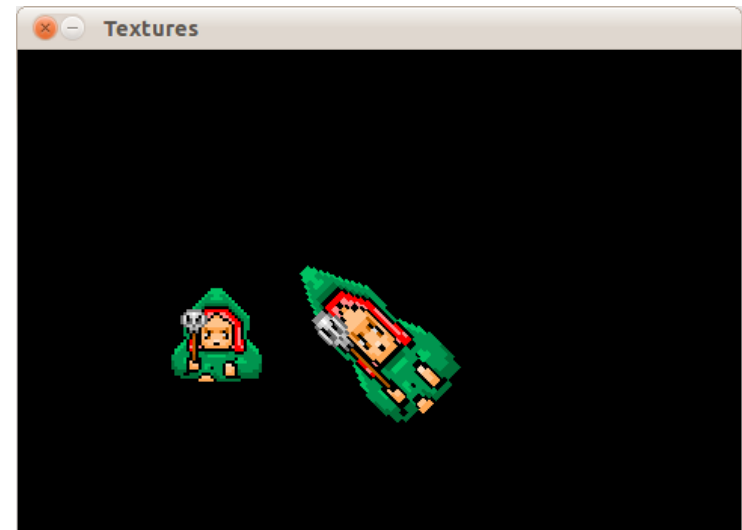
```
public void render() {  
    batch.begin();  
    batch.draw(druidTexture, 100, 100);  
    batch.draw(druidTexture, 132, 132);  
    batch.draw(druidTexture, 164, 164);  
    batch.end();  
}
```



Rotation & Scaling



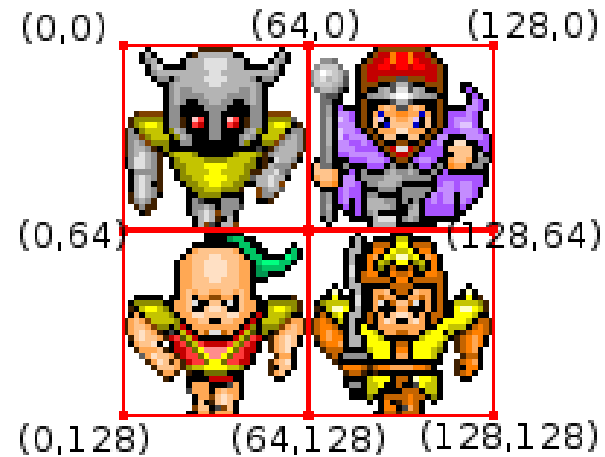
```
public void render() {  
    batch.begin();  
    batch.draw(druidTexture, 100, 100);  
    batch.draw(druidTexture, 200, 100, 32, 32, 64, 64,  
              1f, 2.0f, 45f, 0, 0, 64, 64, false, false);  
    batch.end();  
}
```



TextureRegion



Spritesheet (single image)



Regions with coordinates

TextureRegion

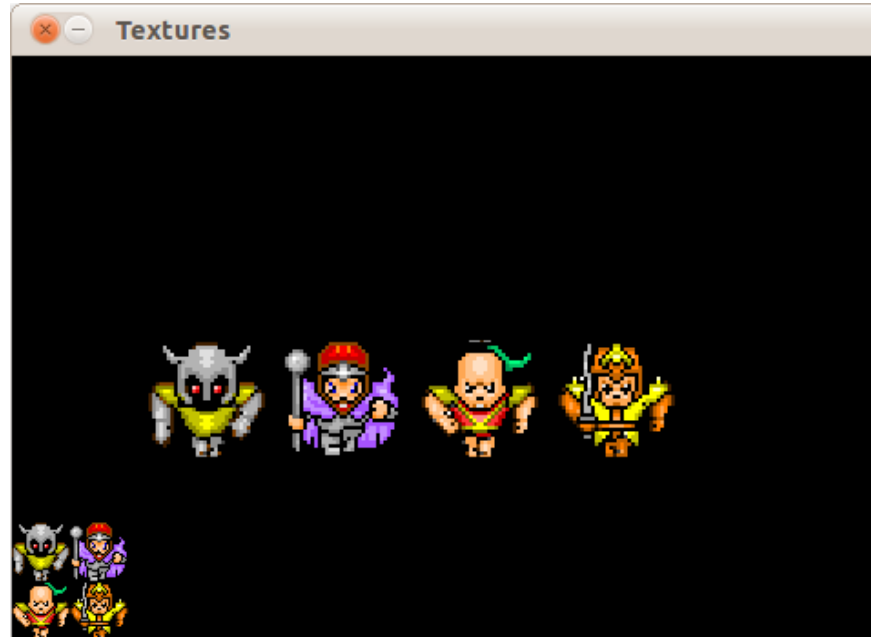


```
public class TextureFun implements ApplicationListener {
    private Texture texture;
    private SpriteBatch batch;
    private TextureRegion[] regions = new TextureRegion[4];

    public void create() {
        texture = new Texture(Gdx.files.internal("sprite_sheet.png"));
        batch = new SpriteBatch();
        regions[0] = new TextureRegion(texture, 0, 0, 64, 64);
        regions[1] = new TextureRegion(texture, 0.5f, 0f, 1f, 0.5f);
        regions[2] = new TextureRegion(texture, 0, 63, 64, 64);
        regions[3] = new TextureRegion(texture, 0.5f, 0.5f, 1f, 1f);
    }

    public void render() {
        batch.begin();
        batch.draw(texture, 0, 0, 64, 64);
        for (int i = 0; i < regions.length; i++)
            batch.draw(regions[i], 75 * (i + 1), 100);
        batch.end();
    }
}
```

TextureRegion



...

`TextureRegion[][] regions = TextureRegion.split(texture, 64, 64)`

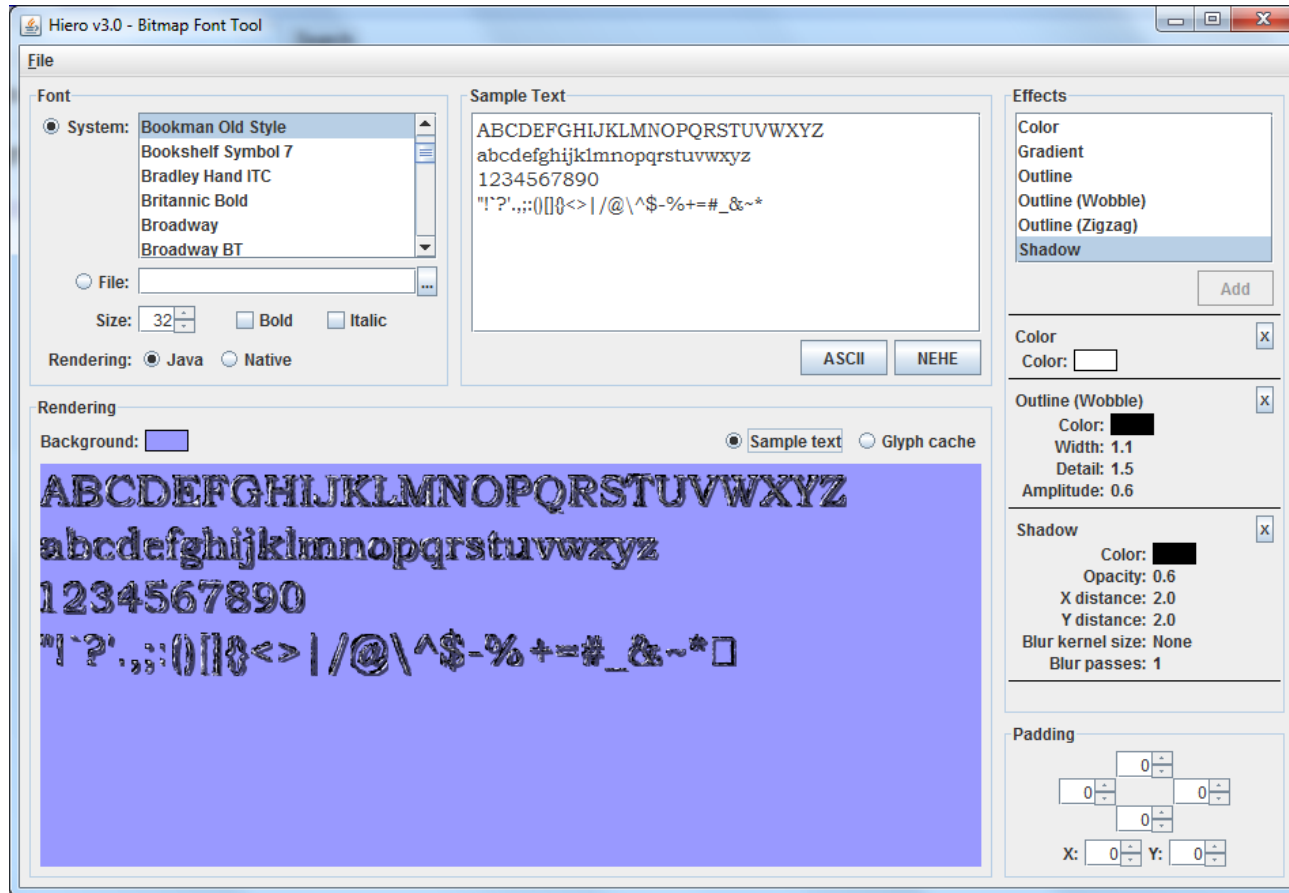
...

Blending & Viewport



- Viewport is determined automatically
 - Mapping pixels to pixels
 - Determined at begin()
- Blending
 - Enabled by default
 - Disable for performance critical things

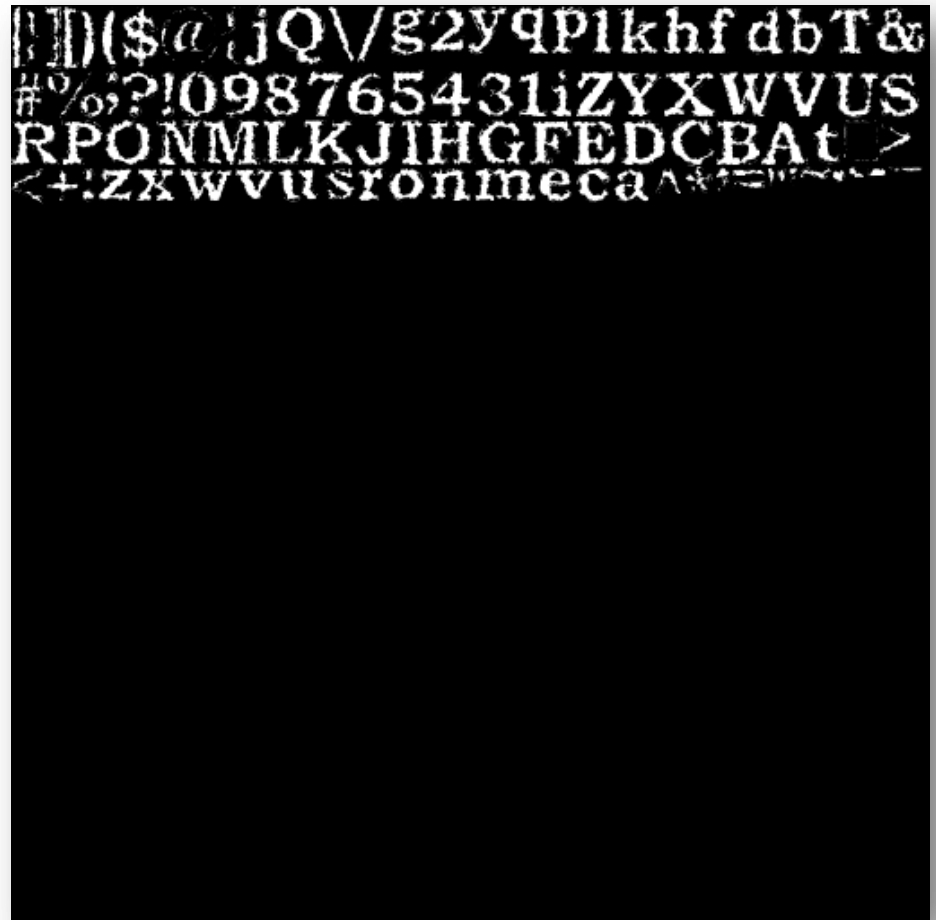
Using Fonts



Hiero Bitmap Font Tool



- Creates 2 files
 - myFont.fnt
 - myFont.png



Using Bitmap Fonts



```
// Member  
BitmapFont font;  
// [...]  
  
// init font from files  
font = new BitmapFont(new FileHandle(new File("myFont.fnt")),  
    new FileHandle(new File("myFont.png")), false);  
// [...]  
  
// draw in the main loop / sprite batch  
font.draw(spriteBatch, "3765 pts.", 5, Gdx.graphics.getHeight() - 5);
```

Input



- Interface to the input facilities
- Allows to poll state of
 - the keyboard
 - touch screen
 - accelerometer
- Event based input available as InputProcessor

Music



- Music interface represents streamed audio
 - Music instance is created with `Audio.newMusic(FileHandle)`
 - Looping, volume, pause and play

Sound



- Sound interface represents in-memory audio clips
 - Feasible for small files
 - Can be played multiple times at once
 - Looping, volume, play and stop

Walkthrough ...



- Memory

What is MS XNA?



- A framework & managed runtime for video game development
- XNA stands for "XNA is Not an Acronym"
- XNA is based on the .NET Compact Framework
- XNA lets you develop for
 - Windows, Windows Phone, Xbox & Zune



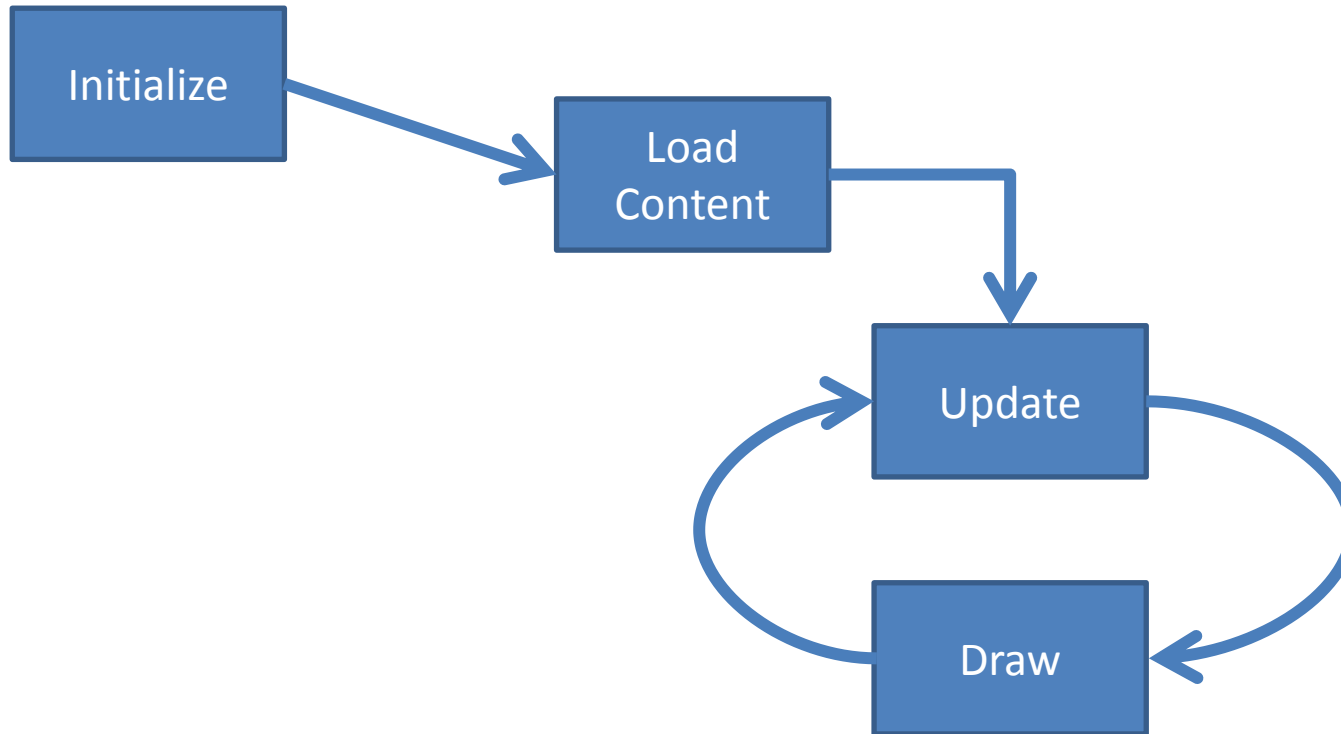
What is MS XNA?



- Integrates with Visual Studio
- C# und VB supported
- Windows Phone Emulator

- Note: MS has ceased development with Feb. 1st 2013

XNA Basic Structure



XNA Basic Structure



```
/// Allows the game to perform any initialization it needs to before starting to run.
```

```
protected override void Initialize()
```

```
/// LoadContent will be called once per game and is the place to load  
/// all of your content.
```

```
protected override void LoadContent()
```

```
/// UnloadContent will be called once per game and is the place to unload all content.
```

```
protected override void UnloadContent()
```

```
/// Allows the game to run logic such as updating the world,  
/// checking for collisions, gathering input, and playing audio.
```

```
protected override void Update(GameTime gameTime)
```

```
/// This is called when the game should draw itself.
```

```
protected override void Draw(GameTime gameTime)
```

Sprites



```
// --< in LoadContent() >--  
// Create a new SpriteBatch  
spriteBatch = new SpriteBatch(GraphicsDevice);  
// Sprite and position  
sprite = Content.Load<Texture2D>("speaker");  
posSprite = new Vector2(graphics.GraphicsDevice.Viewport.Width / 2 -  
    sprite.Width/2,  
    graphics.GraphicsDevice.Viewport.Height - 60 - sprite.Height);  
  
// --< in Draw(GameTime gameTime) >--  
GraphicsDevice.Clear(Color.Black);  
spriteBatch.Begin();  
spriteBatch.Draw(sprite, posSprite, Color.White);  
spriteBatch.End();
```

Sprites



- Rotation
 - `spriteBatch.Draw(sprite, screenpos, null, Color.White, RotationAngle, origin, 1.0f, SpriteEffects.None, 0f)`
 - `origin.x`, `origin.y` give the center of rotation relative to sprite
- Scaling
- Tinting

Drawing Text



```
SpriteFont Font1;  
Vector2 FontPos;  
protected override void LoadContent()  
{  
    spriteBatch = new SpriteBatch(GraphicsDevice);  
    Font1 = Content.Load<SpriteFont>("Courier New");  
    FontPos = new Vector2(graphics.GraphicsDevice.Viewport.Width / 2,  
        graphics.GraphicsDevice.Viewport.Height / 2);  
}
```

- Loading from a true type font.

Drawing text



```
protected override void Draw(GameTime gameTime)
{
    GraphicsDevice.Clear(Color.CornflowerBlue);
    spriteBatch.Begin();
    string output = "Hello World";
    Vector2 FontOrigin = Font1.MeasureString(output) / 2;
    spriteBatch.DrawString(Font1, output, FontPos,
        Color.LightGreen, 0, FontOrigin, 1.0f,
        SpriteEffects.None, 0.5f);
    spriteBatch.End();
    base.Draw(gameTime);
}
```

Getting Input



```
// --< in Update(GameTime gameTime) >--
if (GamePad.GetState(PlayerIndex.One).Buttons.Back ==
    ButtonState.Pressed)
    this.Exit(); // exit game
if (GamePad.GetState(PlayerIndex.One).DPad.Left ==
    ButtonState.Pressed)
    posSprite.X -= 1; // move sprite to the left
if (GamePad.GetState(PlayerIndex.One).DPad.Right ==
    ButtonState.Pressed)
    posSprite.X += 1; // move sprite to the right
// analog input
posSprite.X +=
    GamePad.GetState(PlayerIndex.One).ThumbSticks.Left.X * 5;
```

Sound Effects



```
SoundEffect soundEffect;  
// ...  
soundEffect = Content.Load<SoundEffect>("kaboom");  
// ...  
instance.IsLooped = true; // in case of looping  
soundEffect.Play();  
  
// streaming sound  
soundfile = TitleContainer.OpenStream  
    (@"Content\tx0_fire1.wav");  
soundEffect = SoundEffect.FromStream(soundfile);
```

XNA Additional Features

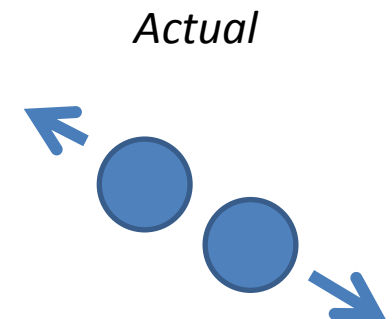
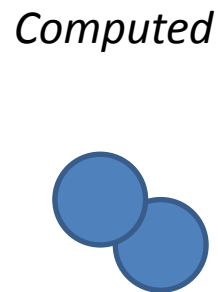
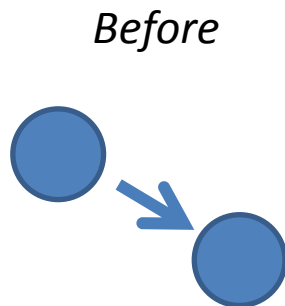


- Xbox Live
 - gamer profiles, presence, avatars
- Networking & multiplayer
 - finding & joining sessions
 - sending & receiving network data
 - network session management
- Packaging & sharing

Collision Detection



- Collision occurs before the overlap
- So an overlap indicates a collision
- Example Pool Billiard



Bounding Boxes



Triangle Mesh



Axis Aligned Bounding Box



Bounding Circle

Triangle Mesh



- Convex hull, convex polygon
- Tight approximation of the object
- Requires storage
- Hard to create
- Expensive to test



Axis Aligned Bounding Box



- Smallest box containing the object
- Edges are aligned to x-axis and y-axis
- Fast to test
- Less precise than triangle mesh
- Stored as corner + width + height



Bounding Circle



- Smallest circle that contains the object
- Very fast test
- Least precise method
- Stored as center + radius



Bounding shape updates



- Every object in the game has
 - position, scale, orientation &
 - bounding shape
- Position, scale and orientation are influenced by the game play
 - Bounding shape has to be update accordingly

Bounding shape updates



- Changes in position
 - Move vertices, box corner or circle center
- Changes in scale
 - Easy if center of change is center of object
 - Scale vectors in relation to center,
 - or change radius of a bounding circle

Bounding shape updates



- Change in orientation

- if center of rotation is center of object ...
- for triangle meshes triangles are rotated
- bounding circles remain untouched
- for bounding boxes corner points have to be rotated & a new box has to be found



Phases in Collision Detection



- Broad phase
 - Identify potentially colliding objects
 - With less precise bounding shapes
- Medium phase
 - Determine actually colliding objects
- Narrow phase
 - Find when (contact time) and where (contact set)

Tools & Frameworks



- FMOD
- RAD Game Tools
- Autodesk Scaleform
- Havok
- (Bullet)
- (Box2D)

FMOD



- **FMOD Ex**
 - Low level API (mixer, interfaces, dsp, 3D, ...)
- **FMOD Event System**
 - Built on FMOD Ex
 - Access to events designed in Designer
- **FMOD Designer**
 - Tool for Event creation
 - Looping, effects, etc.



FMOD Licenses



- Non Commercial License
 - free license
- Commercial License
 - 1,500 – 6,000 \$
- Casual License
 - 500 – 3,000 \$
- Shareware License
 - 100 \$ per platform & product

RAD Game Tools



- Privately held company owned by Jeff Roberts & Mitch Soule
- Based in Kirkland, Washington
- Develops video game software technologies
- Hires one specific person to write, document and support each single product
- refuses to patent any of their technologies



RAD Game Tools



- 1988 Windows dev & consulting
- 1993 Smacker, 8-bit video codec (time of CDs)
- 1994 Fastest growing company in Utah, 63rd in U.S.
- 1995 Aquisition of Miles Sound System
- 1999 Release of Bink
- 2002 Pixomatic (software 3D fallback system)



Bink (2)



- Video game codec for games
 - licensed for > 6,200 games
- Full control through Bink SDK
 - no callbacks etc.
- Self-contained, no 3rd party SW



Bink



- Ressource efficient
 - less memory
 - full dual core, 75% SIMD
- Multiple platforms
 - same API on up to 14 different platforms
- Supported by many game engines
 - eg. already integrated in Unreal Engine



Miles Sound System



- Licensed for > 5,200 games
- Integrates
 - high-level sound authoring with 2D and 3D digital audio
 - featuring streaming, environmental reverb, DSP filtering, multichannel mixing
 - highly-optimized audio decoders (MP3, Ogg and Bink Audio)
- Designer for Windows
- Deployment to “all” platforms supported



Miles Sound System



- MP3 decoder and license
- Bink audio decoder
 - 30% smaller memory footprints
- Ogg Vorb audio decoder



Iggy Game UI



- User Interface library for
 - Windows, Xbox 360, Sony PS3, iOS and Nintendo Wii / U
- Utilizes Flash content (up to v9)
 - includes ActionScript 3 interpreter
- Interactive performance tools
 - for memory and performance debugging



Iggy Game UI



- A triangulator to convert vector shapes to triangles
- A bitmap font rasterizer
- Conversion of wide lines to triangles
- A system to generate edge anti-aliasing the polygons
- Linear and circular gradient support
- Shader based GPU-accelerated implementation of filter effects defined by Flash



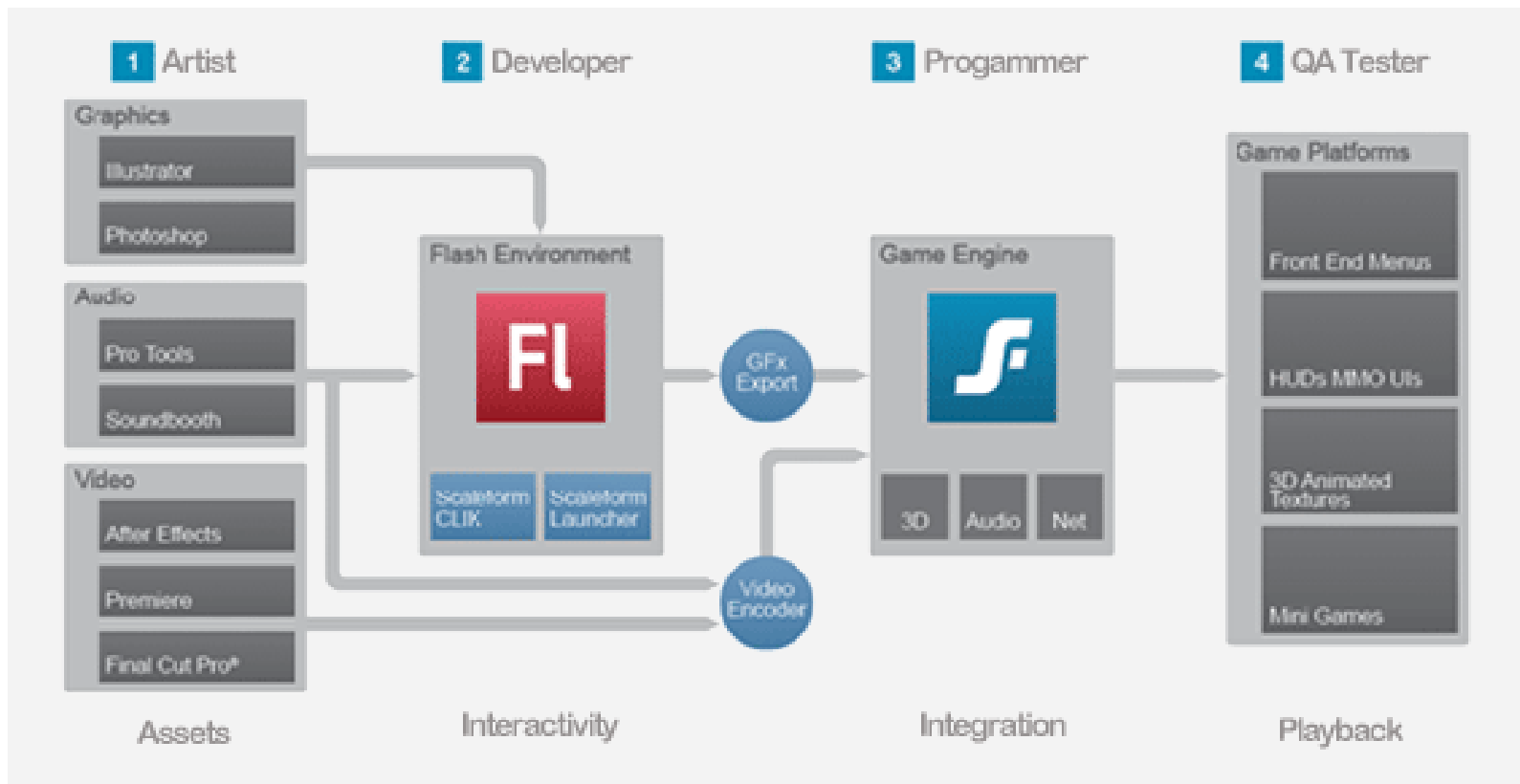
Iggy Game UI



- A full audio mixer
- MP3 decoder from Miles (including patent rights)
- Custom memory management
 - Multiple instances
- Debugging tools:
 - Iggy Explorer (to view the hierarchies of the artists),
 - Telemetry bindings
 - and full memory allocation tracking.



Autodesk Scaleform



Scaleform



1. Create UI assets, including bitmaps, vectors, audio, and video using the Adobe Creative Suite
2. Import assets into Flash Studio and add interactivity and animation
3. Export Flash content to game & connect UI
4. Test the Flash content navigation, localization and functionality running in game

Autodesk®
GAMEWARE 

Havok Physics



- Irish company, since 1998
 - owned by Intel
- Real-time physics simulation
 - collision and dynamics of rigid bodies
 - ragdolls & character animation
 - movement of vehicles

Havok Cloth



- Simulation of cloth, hair, foliage and other deformable objects
- Animate believable behaviors of character garments and environmental cloth
- <http://youtu.be/qT0zOgAdYWk?hd=1>

Havok Destruction



- Tool set & runtime SDK
- Fracture meshes and use in-game
- Destruction of complex structures
- Simulate destruction in runtime

- <http://youtu.be/clcg5eotZlY?hd=1>

Autodesk



- Maya & 3ds Max
- Educational programs
 - Autodesk University
 - Student licenses (<http://students.autodesk.com>)

Autodesk
Education Community

Thanks ...



... any questions?