Game Development 1

Proseminar *Objektorientiertes Programmieren mit .NET und C#*

Philipp Kaiser

Institut für Informatik
Software & Systems Engineering

Agenda

- Allgemeine Spieleentwicklung
- XNA: Vereinte Programmierschnittstellen
- Entführung in die Praxis

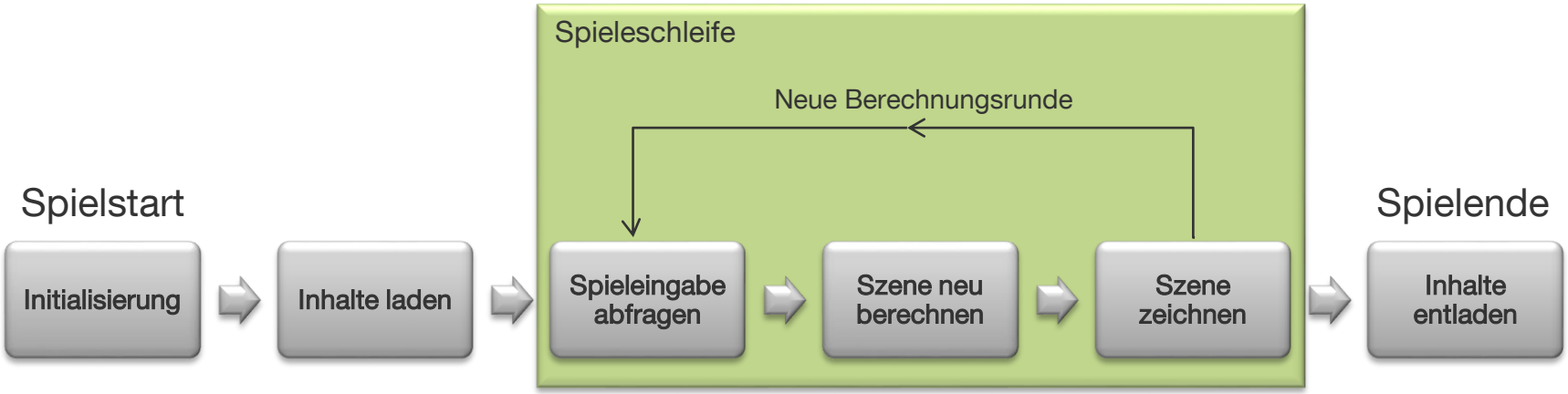
Allgemeine Spieleentwicklung

- Geschichte der Spieleentwicklung
- Das Rundensystem
- Performanz
- Programmierschnittstellen

Geschichte der Spieleentwicklung

- Pong (1972)
- NES(1983)
- Wolfenstein (1992)
- Die Sims (2000)
- World of Warcraft (2004)
- Battlefield 3 (2011)

Das Rundensystem



Perfomanz

- Die Spieleschleife muss für jedes Bild (Frame) einmal durchlaufen werden
- Für ein flüssiges Spielgefühl werden 30 frames per second (fps) benötigt
- Art und Aufwand des Spieles entscheiden über Rechenaufwand der Schleife

Wahl der Programmiersprache

Programmierschnittstellen

- Entwicklung von Grafik-APIs aufgrund der enormen Komplexität der Grafikkarte
- Einfaches und grafikkartenunabhängiges Anzeigen von 2D und 3D-Grafik
- Die zwei bekanntesten sind OpenGL und DirectX

XNA: Vereinte Programmierschnittstellen

- Überblick und Geschichtliches
- Schnittstellen
- Ein neues Projekt
- Die Spieleschleife
- Die Content-Pipeline
- Das Zeichnen
- Input
- Kollision
- 2D und 3D

Überblick und Geschichtliches

XNA (***X****NA's* ***N****ot* ***A****cronymed*) ist eine Technologie zur Spieleentwicklung für Microsoft Windows, Xbox 360, Microsofts MP3-Player Zune sowie Windows Phone 7.

- XNA Game Studio 1.0 erschien am 11. Dezember 2006
- Aktuell: XNA Game Studio 4.0 vom 16. September 2010

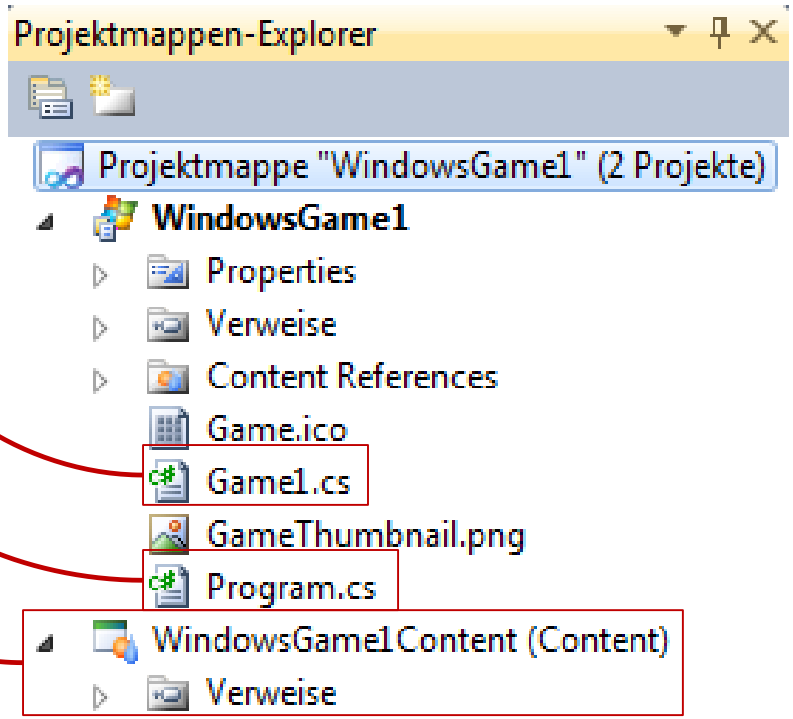
Schnittstellen

- Direct3D (Ausgabe in 2D und 3D)
- XACT (Audioausgabe)
- XInput (Kommunikation mit Input-Geräten)

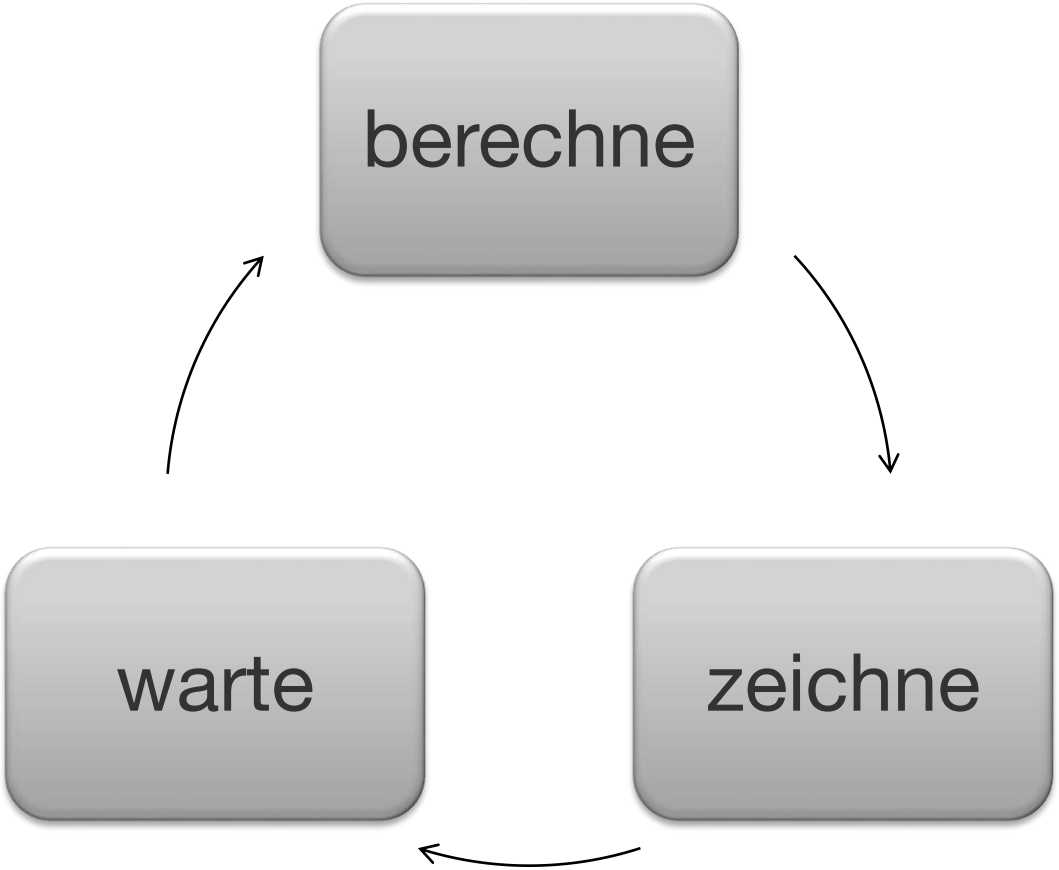
Ein neues Projekt

XNA erstellt automatisch:

- Eine Game-Klasse
- Eine Main-Methode
- Ein Content-Verzeichnis
  für alle Bilder/Texturen,
  Musik, Sound, Schriftarten,
  …



Die Spieleschleife (1)



Die Spieleschleife (2)

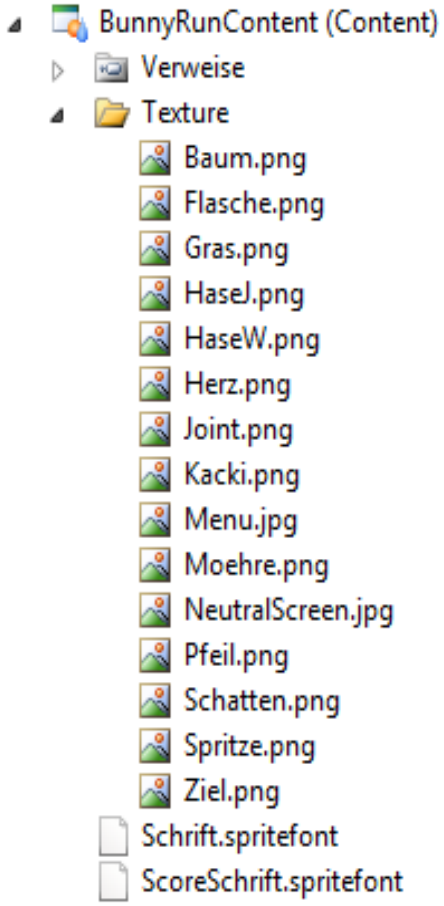
```
public class Game1 : Microsoft.Xna.Framework.Game {
        GraphicsDeviceManager graphics;
        SpriteBatch spriteBatch;

        public Game1()[...]

        protected override void Initialize() [...]

        protected override void LoadContent() [...]

        protected override void UnloadContent() [...]

        protected override void Update(GameTime gameTime)[...]

        protected override void Draw(GameTime gameTime)[...]
}
```

Contentpipeline (1)

- Ein Asset ist eine Ressource für das Spiel, z.B. 3D Modelle, Audiodateien oder Grafiken
- Die Content Pipeline von XNA ermöglicht es einfach Assets für das Spiel zu konvertieren
- XNA bereitet alle Dateien entsprechend dem Zielsystem auf
  (z.B. unteschiedliche Audioformate für Windows und XBox)

Kein Problem mit verschiedenen Formaten

Die Contentpipline (2)




```
[…]

    private Texture2D baumTexture;
    private Texture2D bottleTexture;
    private Texture2D grasTexture;
    private SpriteFont scoreFont;


protected override void LoadContent(){
      baumTexture = Content.Load<Texture2D>("Texture/Baum");
      bottleTexture = Content.Load<Texture2D>("Texture/Flasche");
      grasTexture = Content.Load<Texture2D>("Texture/Gras");
      scoreFont = Content.Load<SpriteFont>("ScoreSchrift");

}

[…]
```

Das Zeichnen

Für 2D Zeichnungen benötigt man die Hilfe des SpriteBatch-Objektes

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

      spriteBatch.Draw(targetTexture, target.position, Color.White);
      spriteBatch.Draw(baumTexture, new Vector2(x, y), Color.White);
      spriteBatch.DrawString(font, "Level:“, new Vector2(5, 25), Color.Black);


      spriteBatch.End();
}
```

Input

Alle Eingaben müssen extra abgefragt werden!

```
GamePadState gamePad = GamePad.GetState(PlayerIndex.One);
KeyboardState keyboard = Keyboard.GetState();
    if (keyboard.IsKeyDown(Keys.Left) && bunny.position.X > 0) {
           bunny.position = new Vector2(bunny.position.X - SPEED *
           (float)gameTime.ElapsedGameTime.TotalSeconds, bunny.position.Y);
    }
    if (keyboard.IsKeyDown(Keys.Right)) {…}
    if (keyboard.IsKeyDown(Keys.Up)) {…}
    if (keyboard.IsKeyDown(Keys.Down)) {…}
    if (keyboard.IsKeyDown(Keys.Space)) {
           bunny.jump = true;
    }
    if (keyboard.IsKeyDown(Keys.Escape)) {
           ingame = false;
           pause = true;
    }
```

Kollision

- XNA bietet eine einfache Art 2D Kollision zu überprüfen.

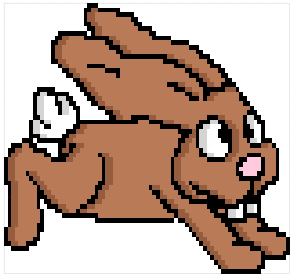
```
Rectangle target = new Rectangle(PositionX1, PositionY1, TextureWidth1, TextureHeight1);
Rectangle hindernis = new Rectangle(PositionX2, PositionY2, TextureWidth2, TextureHeight2);
if (hindernis.Intersects(target)) {
    […]
}
```

- Für 3D Anwendungen bietet XNA die BoundingBox und BoundingSphere
- Sonstige Kollisionsmodelle müssen selbst implementiert werden

2D und 3D

- XNA bietet eine einfache Möglichkeit für 2D Projekte
- 3D Projekte benötigen Modelle und Wissen im Umgang mit Vertices
- Vertexeigenschaften werden von XNA bereitgestellt
- Meist ist ein 3D-Projekt alleine nicht mehr zu bewerkstelligen

Entführung in die Praxis



Fragen?

Vielen Dank für Ihre Aufmerksamkeit!