Höhere Technische Bundeslehranstalt Wien 3, Rennweg

IT & Mechatronik

HTL Rennweg: Rennweg 89b

A-1030 Wien, Tel +43 1 24215-10, Fax DW 18

Diplomarbeit

eventuell KURZTITEL  
Ausgeschriebener Titel der Diplomarbeit

ausgeführt an der  
Höheren Abteilung für  
Informationstechnologie/Ausbildungsschwerpunk  
der Höheren Technischen Lehranstalt Wien 3 Rennweg

im Schuljahr 20??/20??

durch  
Mitarbeiter Eins (alphabetisch)  
Mitarbeiter Zwei  
Mitarbeiter Drei  
Mitarbeiter Vier

unter der Anleitung von

Hauptbetreuer  
eventuell Nebenbetreuer

Wien, Oktober 2014

Kurzfassung

Darum geht es.

Auf der Registerkarte 'Einfügen' enthalten die Kataloge Elemente, die mit dem generellen Layout des Dokuments koordiniert werden sollten. Mithilfe dieser Kataloge können Sie Tabellen, Kopfzeilen, Fußzeilen, Listen, Deckblätter und sonstige Dokumentbausteine einfügen. Wenn Sie Bilder, Tabellen oder Diagramme erstellen, werden diese auch mit dem aktuellen Dokumentlayout koordiniert. Die Formatierung von markiertem Text im Dokumenttext kann auf einfache Weise geändert werden, indem Sie im Schnellformatvorlagen-Katalog auf der Registerkarte 'Start' ein Layout für den markierten Text auswählen.

Text können Sie auch direkt mithilfe der anderen Steuerelemente auf der Registerkarte 'Start' formatieren. Die meisten Steuerelemente ermöglichen die Auswahl zwischen dem Layout des aktuellen Designs oder der direkten Angabe eines Formats. Wählen Sie neue Designelemente auf der Registerkarte 'Seitenlayout' aus, um das generelle Layout des Dokument s zu ändern. Verwenden Sie den Befehl zum Ändern des aktuellen Schnellformatvorlagen-Satzes, um die im Schnellformatvorlagen-Katalog verfügbaren Formatvorlagen zu ändern.

Abstract

That‘s why.

Ensure that the entire document's proofing language is set to English. Remember that proofing language is set for each word, even each individual character, not for the document as a whole, so even a single character in the wrong place can throw it off.

Open the document and hit Ctrl+A to select all text and objects in the document. Then go to Review | Language | Set Proofing Language and ensure it's set to "English

Ehrenwörtliche Erklärung

Ich versichere,

* dass ich meinen Anteil an dieser Diplomarbeit selbstständig verfasst habe,
* dass ich keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe
* und mich auch sonst keiner unerlaubten Hilfe bzw. Hilfsmittel bedient habe.

Wien, am

<eigenhändige Unterschriften aller Teammitglieder>

Präambel

Die Inhalte dieser Diplomarbeit entsprechen den Qualitätsnormen für „Ingenieurprojekte“ gemäß § 29 der Verordnung des Bundesministers für Unterricht und kulturelle Angelegenheiten über die Reife- und Diplomprüfung in den berufsbildenden höheren Schulen, BGBl. Nr. 847/1992, in der Fassung der Verordnungen BGBl. Nr. 269/1993, Nr. 467/1996 und BGBl. II Nr. 123/97.

Liste der betreuenden Lehrer

<[Dir|AV|Prof], akad. Grad, Vorname Name Hauptbetreuer>

<[Dir|AV|Prof], akad. Grad, Vorname Name Hauptbetreuer Stellvertreter>

<[Dir|AV|Prof], akad. Grad, Vorname Name Betreuer> ... (in alphabetischer Reihenfolge des Nachnamens)

<[Dir|AV|Prof], akad. Grad, Vorname Name Betreuer>

Liste der Kooperationspartner:

Inhaltsverzeichnis

[Kurzfassung 2](#_Toc401212867)

[Abstract 3](#_Toc401212868)

[Ehrenwörtliche Erklärung 4](#_Toc401212869)

[Präambel 5](#_Toc401212870)

[1 Beschreibung der Formatierung 7](#_Toc401212871)

[1.1 Vorlagen 7](#_Toc401212872)

[1.1.1 Formatvorlagen 7](#_Toc401212873)

[1.1.2 Schriften und Absätze 7](#_Toc401212874)

[1.1.3 Bilder 8](#_Toc401212875)

[1.1.4 Tabellen 9](#_Toc401212876)

[1.1.5 Formel 10](#_Toc401212877)

[1.1.6 Sourcecode 11](#_Toc401212878)

[1.1.7 Fachbegriffe 12](#_Toc401212879)

[1.1.8 Zitieren 12](#_Toc401212880)

[1.1.9 Quellenverzeichnis 13](#_Toc401212881)

[1.1.10 Rechtliches zum Zitieren 13](#_Toc401212882)

[1.2 Bad Practice 14](#_Toc401212883)

[1.3 Beispiele 16](#_Toc401212884)

[1.3.1 Zitieren mit Latex 16](#_Toc401212885)

[1.3.2 Formatierungen 16](#_Toc401212886)

[Tabellenverzeichnis 18](#_Toc401212887)

[Abbildungsverzeichnis 19](#_Toc401212888)

[Stichwortverzeichnis 20](#_Toc401212889)

# Software

Alle Softwarekomponente des Spiels wurden in C# innerhalb des Monogame Frameworks geschrieben.

## Monogame Framework

Abbildung 1. Monogame Logo  
Quelle: https://upload.wikimedia.org/wikipedia/commons/e/e6/MonoGame\_Logo.svg

Monogame ist eine Open-Source Weiterführung vom von Microsoft entwickelten XNA’ Framework. Es basiert auf der Programmiersprache C# und unterstützt DirectX als auch OpenGL.

* DirectX ist eine von Microsoft entwickelte Graphik API. Sie wird von Windows Systemen und Xbox Konsolen unterstützt.
* OpenGL ist eine Open-Source Graphik API. Sie ist Platform übergreifend und kann für Windows, macOS und Linux Systeme verwendet geben.
* Xamarin?

Diese Bibliotheken werden vom Monogame Framework zum kommunizieren mit der Graphikkarte verwendet. Beim Erstellen eines Projektes muss man sich für eines dieser Zwei entscheiden.

Für dieses Projekt bietet sich OpenGL am besten an, da wir von keinen relevanten Einschränkungen betroffen sind und wir dann mit unserem Produkt mehrere Platformen gleichzeitig ansprechen können.

Eine neu erstellte Monogame Vorlage sieht folgender Maßen aus:

|  |
| --- |
| using Microsoft.Xna.Framework;  using Microsoft.Xna.Framework.Graphics;  using Microsoft.Xna.Framework.Input;  namespace Game1  {  public class Game1 : Game  {  GraphicsDeviceManager graphics;  SpriteBatch spriteBatch;    public Game1()  {  graphics = new GraphicsDeviceManager(this);  Content.RootDirectory = "Content";  }  protected override void Initialize()  {  base.Initialize();  }  protected override void LoadContent()  {  spriteBatch = new SpriteBatch(GraphicsDevice);  }  protected override void UnloadContent()  {  }  protected override void Update(GameTime gameTime)  {  if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed || Keyboard.GetState().IsKeyDown(Keys.Escape)) Exit();  base.Update(gameTime);  }  protected override void Draw(GameTime gameTime)  {  GraphicsDevice.Clear(Color.CornflowerBlue);  base.Draw(gameTime);  }  }  } |

Initialize() wird für das Laden aller Elemente verwendet, welche nicht von der Pipeline geladen werden. (Services, Spielstände, Datenbanken)

LoadContent() wird für das Laden aller Elemente verwendet, welche von der Pipeline geladen werden. (Graphiken, Sounds, Schriften)

UnloadContent() wird für das Entladen aller Elemente verwendet, welche von der Pipeline geladen wurden. Dies befreit die verbrauchte RAM.

Update() wird für das Aktualisieren der Spiellogik verwendet.

Draw() wird für das Zeichnen mit der Graphikkarte verwendet.

Initialize() und LoadContent() werden immer beim Programmstart aufgerufen.

UnloadContent() wird beim Programmende aufgerufen.

Update() und Draw() werden ständig aufgerufen und bilden die GameUpdateLoop.

Beim ausführen dieses Programmes wird ein Bildschirm mit einem hellblauen Hintergrund erstellt. Beim Drücken der ESCTaste oder dem Menüknopf eines Controllers wir das Fenster geschlossen. Diese Vorlage ist alles was Monogame von uns an Arbeit entnimmt, alles Andere muss selbst entwickelt werden.

### UpdateLoop

Die Funktionen Update() und Draw() bilden die UpdateLoop, welche jede Bildwiederholung aufgerufen wird. Logikcode ist hier vom Graphikcode getrennt. Dies hat best practice, als auch technische Gründe (da die Aufrufung dieser zwei Methoden sich unter Extremfällen doch verschieden verhält, ist für uns aber nicht relevant).

Update() {

In der Update Funktion wird der Logikcode ausgeführt. Dies beinhaltet so ziemlich alles, was nicht direkt auf den Bildschirm gezeichnet werden muss. Jeder zeitabhängige Code verwendet die DeltaTime zu der Berechnung von Zeitdifferenzen. Der Großteil des Codes wird sich hier befinden.

}

Draw() {

In der Draw Funktion wird der Code zum Zeichnen ausgeführt. Dies wird mithilfe eines Spritebatches erledigt. Ein Spritebatch ist eine Ansammlung von Befehlen die an die Graphikkarte am ende eines Updateloops geschickt werden. Diese werden innerhalb eines Batches organisiert, um die Optimierung von z.B. sich wiederholenden Graphiken zu ermöglichen.

}

//Siehe Kapitel Graphiken für mehr über spritebatches undso

### DeltaTime

Wenn man z.B. Bewegung simulieren möchte, benötigt man einen Bezug zu der vergangenen Zeit. Diese Funktionalität vergibt uns das GameTime Objekt, welches von Monogame bereitgestellt und verwendet wird.

Das Objekt GameTime besteht aus folgenden Attributen:

* TimeSpan TotalGameTime Die vergangene Zeit seit Spielstart
* TimeSpan ElapsedGameTime Die vergangene Zeit seit letztem Update()
* Bool IsRunningSlowly Ob das Spiel unter der festgelegten Frequenz rennt

Bei der Berechnung von Zeit gibt es zwei Herangehensweisen:

Fixe Wiederholrate

Bei der fixen Wiederholrate wird die Frequenz mit der sich das Spiel selbst aufruft limitiert.

Es erlaubt für einfachere Kalkulation von Zeitdifferenzen, da diese im Idealfall immer gleich sind. Also wenn man sein Spiel auf 60fps’ limitiert, wäre ein Zeitsprung zwischen zwei Bildern immer 0.0166s (1/60s) lang. Damit umgeht man Berechnungen mithilfe von ElapsedGameTime. Wenn es dazu kommt, dass die Wiederholrate durch Performanceprobleme unter diese Schwelle fällt, verhält sich die Spiellogik gleich, da die Zeitdifferenz statisch definiert wurde.

Die Vor- und Nachteile diese Methode sind die Folgende:

+ Zeitberechnung statisch und Logikcode ist vorhersehbarer

+ Performanceschwankungen oberhalb der Schwelle werden abgeschnitten

+ Das System überarbeitet sich nicht, was sinnvoll bei Laptops und Mobilgeräten ist, oder bei limitierter Stromversorgung

- Monitore mit Bildfrequenzen über der Schwelle werden nicht ausgenutzt, kommt negativ bei der Kundschaft an

- Unbrauchbar für Online Multiplayer – durch verschiedener Hardware würde es zu Asynchronisierungen kommen

Diese Methode ist mitlerweile veraltet, wird aber dennoch bei kleineren Projekten gerne verwendet.

Variable Wiederholrate

Bei der variablen Wiederholrate wird die Frequenz nicht limitiert und hat somit kein Maximum, das sie erreichen kann.

Die Vor- und Nachteile diese Methode sind die Folgende:

+ Zeitberechnung statisch und Logikcode ist vorhersehbarer

+ Performanceschwankungen oberhalb der Schwelle werden abgeschnitten

+ Das System überarbeitet sich nicht, was sinnvoll bei Laptops und Mobilgeräten ist, oder bei limitierter Stromversorgung

- Monitore mit Bildfrequenzen über der Schwelle werden nicht ausgenutzt, kommt negativ bei der Kundschaft an

- Unbrauchbar für Online Multiplayer – durch verschiedener Hardware würde es zu Asynchronisierungen kommen