**Dokumentation des Projekts**

**„Virtueller Pflanzenpfleger“**

**1. Deckblatt**

* **Titel des Projekts:** Virtueller Pflanzenpfleger
* **Name des Auszubildenden:** Martin Braun
* **Ausbildungsberuf:** Fachinformatiker/in für Systemintegration
* **Name des Unternehmens:** Lutz & Grub
* **Datum der Abgabe:** 23.10.2024

**2. Inhaltsverzeichnis**

1. Deckblatt............................................................
2. Projektbeschreibung..................................................
3. Projektplanung.......................................................
4. Durchführung..........................................................
5. Ergebnisse.............................................................
6. Fazit..................................................................
7. Tagebuch.................................................................
8. Schritt-für-Schritt-Zusammenfassung des Codes..

**3. Einleitung**

In diesem Projekt habe ich ein Spiel entwickelt, das als virtueller Pflanzenpfleger fungiert. Das Spiel erlaubt es dem Nutzer, eine Pflanze zu gießen, ihr Sonnenlicht zu geben, sie zu düngen und Unkraut zu entfernen. Ziel des Projekts war es, meine Programmierkenntnisse in C# zu vertiefen und ein spaßiges Programm zu erstellen.

**Zielsetzung und Motivation**

Die Hauptmotivation hinter diesem Projekt war es, eine interaktive Anwendung zu erstellen, die den Nutzern das Konzept der Pflanzenpflege näherbringt. Dabei sollte die Nutzererfahrung durch verschiedene Funktionen und Erfolge gefördert werden.

**Relevanz des Projekts im Kontext der Ausbildung**

Das Projekt hat mir ermöglicht, praktische Erfahrungen in der Softwareentwicklung zu sammeln, insbesondere in der Anwendung von Konsolenprogrammierung und der Nutzung von Timern und Zufallszahlen in C#.

**4. Projektbeschreibung**

Das Spiel „Virtueller Pflanzenpfleger“ simuliert die Pflege einer Pflanze, die in verschiedenen Umgebungsbedingungen wachsen kann. Es gibt mehrere Interaktionen, die der Nutzer durchführen kann, um das Wachstum der Pflanze zu fördern und sie gesund zu halten.

**Technische und organisatorische Rahmenbedingungen**

* **Sprache:** C#
* **Entwicklungsumgebung:** Visual Studio
* **Zielplattform:** Windows

**Verwendete Technologien, Tools und Methoden**

* Programmiersprache: C#
* IDE: Visual Studio
* Nutzung von objektorientierter Programmierung
* Timer für zeitabhängige Ereignisse

**5. Projektplanung**

**Meilensteinen**

* Projektidee und Konzeptentwicklung
* Grundgerüst der Anwendung erstellen
* Implementierung der Pflanze und ihrer Eigenschaften
* Implementierung der Benutzerinteraktionen
* Tests und Fehlerbehebung

**Ressourcenplanung**

* **Materialien:** Computer, Visual Studio, Google.de, GitHUB

**Risikomanagement**

* **Mögliche Risiken:** Technische Schwierigkeiten bei der Implementierung von Funktionen
* **Handhabung:** Regelmäßige Tests während der Entwicklung, um Probleme frühzeitig zu erkennen

**6. Durchführung**

Die Umsetzung des Projekts verlief in mehreren Schritten. Zu Beginn wurde die grundlegende Struktur des Programms entwickelt, gefolgt von der Implementierung der Logik für die Pflanzenpflege.

**Tägliche Protokolle**

* Konzept entwickelt, erste Skizzen gemacht.
* Grundlegende Klassen erstellt und Logik für die Pflanze implementiert.
* Benutzerinteraktionen programmiert.
* Spieltests durchgeführt.
* Letzte Anpassungen und Dokumentation erstellt.

**Herausforderungen und Probleme**

Eine der größten Herausforderungen war die Implementierung der Timer, die für die zeitabhängigen Ereignisse zuständig sind. Die Lösung bestand darin, die Timer-Ereignisse sorgfältig zu planen und zu testen.

**Änderungen am ursprünglichen Plan**

Ursprünglich war ein komplexerer Shop mit mehr Kaufoptionen geplant. Aufgrund von Zeitmangel wurde dies jedoch vereinfacht, um die Kernfunktionen des Spiels zu optimieren.

**7. Ergebnisse**

Das Ergebnis des Projekts ist ein funktionierendes Spiel, in dem der Nutzer aktiv die Pflege einer Pflanze übernehmen kann. Die Interaktionen sind intuitiv gestaltet und die Erfolge bieten einen zusätzlichen Anreiz, sich um die Pflanze zu kümmern.

**Reflexion über die Erreichung der Projektziele**

Die Projektziele wurden größtenteils erreicht. Die Anwendung funktioniert stabil und bietet eine ansprechende Nutzererfahrung.

**8. Fazit**

Insgesamt habe ich durch dieses Projekt viel über die Programmierung in C# gelernt. Die Arbeit an dem Spiel hat mir nicht nur technische Fähigkeiten vermittelt, sondern auch meine Problemlösungsfähigkeiten verbessert.

**Persönliche Reflexion**

Während der Arbeit an diesem Projekt habe ich festgestellt, dass ich gerne interaktive Anwendungen entwickle. Dies hat meine Motivation, in diesem Bereich weiterzuarbeiten, erheblich gesteigert.

**Ausblick auf mögliche Weiterentwicklungen**

In Zukunft könnte das Spiel um weitere Funktionen wie verschiedene Pflanzenarten und Wetterbedingungen erweitert werden. Auch die Möglichkeit, das Spiel für mobile Plattformen zu adaptieren, wäre denkbar.

**9. Tagebuch**

**Dokumentation des Projekts: Virtueller Pflanzenpfleger in C#**

**Zeitraum der Entwicklung: 15.10.2024 - 22.10.2024**

**15.10.2024**

**09:00 Uhr - Projektinitialisierung**

Heute habe ich das Projekt in Visual Studio erstellt. Ich habe eine Konsolenanwendung gewählt, da das Projekt den Vorgaben entspricht. Ich habe die grundlegenden Klassen und Methoden definiert, die für die Funktionalität des Spiels notwendig sind

**11:00 Uhr - Implementierung der Klasse Plant**

Ich habe die Klasse Plant erstellt, die die Eigenschaften der Pflanze enthält, wie WaterLevel, SunlightLevel und NutrientLevel. Dies war eine interessante Herausforderung, da ich lernen musste, wie man Eigenschaften in C# korrekt implementiert.

**Eigenschaften**:.WaterLevel, SunlightLevel, NutrientLevel, WeedLevel, GrowthStage, und PusteblumenCount: Diese Integer-Werte repräsentieren verschiedene Pflegefaktoren und den aktuellen Zustand der Pflanze. Erfolge (WaterAchievement, SunlightAchievement, etc.): Boolean-Werte zur Überprüfung von bestimmten Meilensteinen in der Pflanzenpflege.

**14:00 Uhr - Überprüfung auf Fehler**

Beim Kompilieren des Codes habe ich einige Fehler in der Syntax festgestellt. Insbesondere gab es Probleme mit den Zugriffsmodifikatoren. Durch das Studium der C# Dokumentation konnte ich diese Fehler beheben. Grundlegende Systemfunktionen (System) und Timer-Funktionen (System.Timers), um Timer für Wetter- und Unkrautereignisse zu nutzen.

**16:00 Uhr - Basislogik implementiert**

Ich habe die grundlegenden Methoden zum Gießen, Düngen und Entfernen von Unkraut erstellt. Ich musste sicherstellen, dass die Pflanzenlevel korrekt aktualisiert werden. Es war frustrierend, als ich merkte, dass ich einige Variablen nicht richtig initialisiert hatte, aber nach einigen Tests konnte ich das Problem beheben.

Water(): Erhöht den Wasserstand und überprüft, ob Erfolge erreicht wurden.

GiveSunlight(): Erhöht das Sonnenlichtniveau und überprüft Erfolge.

Fertilize(): Erhöht den Nährstoffstand und überprüft Erfolge.

RemoveWeeds(): Setzt das Unkrautniveau auf null und überprüft Erfolge.

Grow(): Lässt die Pflanze wachsen, wenn bestimmte Bedingungen erfüllt sind, und gibt Feedback über das Wachstum.

**16.10.2024**

**10:00 Uhr - Wettermechanik**

Ich habe einen Timer implementiert, der alle 20 Sekunden das Wetter ändert. Dabei habe ich gelernt, wie man mit Timern in C# umgeht. Ein kleiner Fehler war, dass ich die Timer nicht richtig gestartet hatte, aber ich konnte das Problem nach einem kurzen Debugging lösen.

**13:00 Uhr - Unkrautwuchs**

Zusätzlich zur Wettermechanik habe ich ein System für das Unkrautwachstum eingeführt, das alle 30 Sekunden aktualisiert wird. Ich habe viel Zeit mit dem Debugging verbracht, da ich nicht sofort die richtige Logik für das Unkrautlevel hatte.

weedTimer: Timer für das Wachstum von Unkraut.

GrowWeeds(): Erhöht das Unkrautniveau alle 30 Sekunden und hat eine Chance, Pusteblumen hinzuzufügen.

**15:00 Uhr - Integration der Erfolge**

Ich habe die Erfolge (Achievements) implementiert. Hierbei stieß ich auf ein Problem mit der Logik, die überprüft, ob die Bedingungen erfüllt sind. Durch das Hinzufügen einer Checkliste konnte ich die Erfolge effizienter verwalten.

Status(): Zeigt den aktuellen Status der Pflanze an.

IsDead(): Überprüft, ob die Pflanze tot ist.

CelebrateGrowth(): Gibt spezielle Nachrichten aus, je nach Wachstumsstufe.

**17.10.2024**

**09:30 Uhr - Visualisierung der Erfolge**

Ich habe das System geändert, um die Erfolge in der Konsole mit verschiedenen Farben darzustellen. Dabei lernte ich Console.ForegroundColor und Console.ResetColor zu verwenden. Dies machte die Ausgabe visuell ansprechender.

**11:00 Uhr - Patch für Wetterwechsel**

Ich habe den ersten Patch für den Wetterwechsel eingeführt, um die Wetterbedingungen in grüner Farbe anzuzeigen. Das war eine Herausforderung, da ich zuerst die Farben falsch gesetzt hatte und die Ausgabe nicht wie gewünscht war.

weatherTimer: Timer für Wetteränderungen.

**15:30 Uhr - Debugging der Wetterwechsel-Logik**

Beim Testen der Wetterwechsel-Logik stellte ich fest, dass die Zufallszahlen nicht gleichmäßig verteilt waren. Nach einigen Anpassungen konnte ich die Logik so optimieren, dass jede Wetterbedingung gleichmäßig auftrat.

**18.10.2024**

**10:00 Uhr - Einführung des Dropsystems**

Ich habe ein Dropsystem implementiert, das Pusteblumen als Belohnung für das Pflegen der Pflanze gibt. Dies erforderte, dass ich die Logik für den Zufallszahlengenerator verstand und implementierte.

**12:30 Uhr - Optimierung des Growth-Systems**

Ich habe die Wachstumsmethoden verbessert, um sicherzustellen, dass die Pflanzen nur wachsen, wenn alle Bedingungen erfüllt sind. Hierbei hatte ich anfangs Probleme mit der Bedingungslogik, da ich nicht alle Statuswerte korrekt überprüfte.

**14:00 Uhr - Shop-Funktionalität**

Ich implementierte einen Shop, in dem Spieler Pusteblumen gegen Boosts eintauschen können. Das Testen des Shop-Systems war herausfordernd, da ich sicherstellen musste, dass die richtigen Werte abgezogen wurden.

AddPusteblumen(int count) und RemovePusteblumen(int count): Methoden zum Hinzufügen und Entfernen von Pusteblumen.

ShowChecklist(): Zeigt die erreichten Erfolge des Benutzers an.

Shop(): Erlaubt dem Benutzer, im Shop Boosts für Wasser, Licht und Dünger zu kaufen, basierend auf der Anzahl der Pusteblumen.

**19.10.2024**

**09:00 Uhr - Patch für achievements**

Ich habe einen Patch erstellt, um die Anzeige der Erfolge zu verbessern. Anstatt sie sofort anzuzeigen, habe ich die Anzeige auf alle erreichten Erfolge umgestellt. Es war frustrierend, die Zeitsteuerung zu implementieren, aber letztendlich habe ich es mit einer einfachen Delay-Methode gelöst.

CheckAchievements(): Überprüft und gibt Erfolge aus, wenn bestimmte Werte erreicht werden.

**11:30 Uhr - Fehlerbehebung**

Ich stieß auf Probleme bei der Darstellung der Erfolge und der Implementierung der Wartezeit zwischen den Anzeigen. Nach vielen Tests konnte ich die Logik korrekt umsetzen.

CheckPlantStatus(): Reduziert Wasser- und Nährstofflevels basierend auf Wetterbedingungen und überprüft den Status der Pflanze.

**20.10.2024**

**10:00 Uhr - Einführung von Tornado-Mechanik**

Ich habe die Tornado-Mechanik implementiert, die zufällig auftritt und die Werte der Pflanze zerstört. Es war eine Herausforderung, die Wahrscheinlichkeit für den Tornado realistisch zu gestalten bei unter 50% ist er so Gut wie nie gekommen über 50% würde das Balancen schwierig.

**13:00 Uhr - Optimierung der Überprüfungsmethoden**

Ich verbesserte die Methoden zur Überprüfung des Pflanzenstatus, um sicherzustellen, dass alle Statuswerte regelmäßig aktualisiert werden. Das war wichtig, um das Spiel fair und herausfordernd zu halten.

**21.10.2024**

**09:00 Uhr - Überarbeitung der Nutzerinteraktion**

Ich verbesserte die Nutzerinteraktion, indem ich die Menüoptionen klarer strukturiert habe. Dies half, die Benutzerfreundlichkeit des Spiels zu steigern und ich bin von Consolwrite auf Consolkey übergegangen macht das spiel einfacher und angenehmer.

**11:00 Uhr - Schlusskorrekturen**

Ich habe eine vollständige Durchsicht des Codes vorgenommen, um sicherzustellen, dass alles korrekt funktionierte. Dabei habe ich viele kleine Fehler gefunden, die das Spielerlebnis beeinträchtigt hätten.

**22.10.2024**

**09:00 Uhr - Endgültige Tests**

Ich führte umfassende Tests durch, um sicherzustellen, dass das gesamte Spiel einwandfrei läuft. Dabei stellte ich sicher, dass die Timer und die Benutzeroberfläche fehlerfrei funktionierten.

**12:00 Uhr - Dokumentation fertiggestellt**

Ich habe die gesamte Dokumentation erstellt und das gesamte Projekt dokumentiert. Dies umfasste die täglichen Fortschritte, die Herausforderungen und die Lösungen, die ich gefunden habe.

**15:00 Uhr - Abschluss**

Ich bin stolz auf das Ergebnis und das Wissen, das ich in den letzten zwei Wochen über C# und die Spielentwicklung gewonnen habe. Das Projekt hat mir wertvolle Erfahrungen gebracht, und ich freue mich darauf, in Zukunft weitere Projekte zu entwickeln.

**Zusammenfassung der Patches**

**Patch Wetterwechsel:** Wetterwechsel wird jetzt in grüner Farbe angezeigt, Unkraut in rot und Tornados ebenfalls in rot. Level-Ups erscheinen in gelber Farbe.

**Patch Tornado:** Einführung eines Tornados, der zufällig auftritt und negative Auswirkungen auf die Pflanze hat.

**Achievement Patch**: Änderungen in der Anzeige der Erfolge, die nun nach einer Minute erneut angezeigt werden.

**Patch Checkliste:** Einführung einer Checkliste zur Überprüfung der Erfolge, die mir half, Bugs zu vermeiden und die Logik zu verbessern.

**Patch Dropsystem:** Einführung eines Dropsystems, das Items wie Pusteblumen generiert.

**Patch Shop:** Ein Shop, in dem Spieler Boosts für ihre Pflanzen kaufen können, wurde implementiert.

\*\***Diese Dokumentation beschreibt die täglichen Fortschritte, die Herausforderungen und die Lösungen, die ich während der Entwicklung des Projekts gefunden habe. Ich hoffe, dass dies hilfreich ist, um die Entwicklung des Spiels zu verstehen und meine Erfahrungen als Schüler, der C# lernt, widerzuspiegeln.**

**Hier ist eine Schritt-für-Schritt-Zusammenfassung des Codes, der Code ist eine einfache Simulation eines virtuellen Pflanzenpflegers in C#. Er umfasst eine Plant-Klasse mit verschiedenen Eigenschaften und Methoden zur Pflege einer Pflanze sowie eine Program-Klasse, die die Hauptlogik steuert.**

**Schritt-für-Schritt Zusammenfassung:**

1. **Imports**:
   * Der Code importiert grundlegende Systemfunktionen (System) und Timer-Funktionen (System.Timers), um Timer für Wetter- und Unkrautereignisse zu nutzen.
2. **Plant-Klasse**:
   * **Eigenschaften**:
     + Name: Name der Pflanze.
     + WaterLevel, SunlightLevel, NutrientLevel, WeedLevel, GrowthStage, und PusteblumenCount: Diese Integer-Werte repräsentieren verschiedene Pflegefaktoren und den aktuellen Zustand der Pflanze.
     + Erfolge (WaterAchievement, SunlightAchievement, etc.): Boolean-Werte zur Überprüfung von bestimmten Meilensteinen in der Pflanzenpflege.
   * **Methoden**:
     + Water(): Erhöht den Wasserstand und überprüft, ob Erfolge erreicht wurden.
     + GiveSunlight(): Erhöht das Sonnenlichtniveau und überprüft Erfolge.
     + Fertilize(): Erhöht den Nährstoffstand und überprüft Erfolge.
     + RemoveWeeds(): Setzt das Unkrautniveau auf null und überprüft Erfolge.
     + Grow(): Lässt die Pflanze wachsen, wenn bestimmte Bedingungen erfüllt sind, und gibt Feedback über das Wachstum.
     + CelebrateGrowth(): Gibt spezielle Nachrichten aus, je nach Wachstumsstufe.
     + Status(): Zeigt den aktuellen Status der Pflanze an.
     + IsDead(): Überprüft, ob die Pflanze tot ist.
     + CheckAchievements(): Überprüft und gibt Erfolge aus, wenn bestimmte Werte erreicht werden.
     + AddPusteblumen(int count) und RemovePusteblumen(int count): Methoden zum Hinzufügen und Entfernen von Pusteblumen.
3. **Program-Klasse**:
   * **Timer**:
     + weatherTimer: Timer für Wetteränderungen.
     + weedTimer: Timer für das Wachstum von Unkraut.
   * **Hauptmethode (Main)**:
     + Erstellt eine neue Pflanze und initialisiert die Timer.
     + Zeigt ein Menü an, in dem der Benutzer verschiedene Befehle zur Pflege der Pflanze eingeben kann (Wasser, Licht, Dünger, etc.).
     + Eine Endlosschleife liest Tasteneingaben des Benutzers und führt die entsprechenden Methoden der Plant-Klasse aus.
     + Überprüft, ob die Pflanze tot ist und stoppt die Timer, wenn dies der Fall ist.
   * **Wetteränderungen**:
     + ChangeWeather(): Zufällige Wetteränderungen (sonnig, bewölkt, regnerisch) werden alle 20 Sekunden simuliert.
   * **Unkrautwachstum**:
     + GrowWeeds(): Erhöht das Unkrautniveau alle 30 Sekunden und hat eine Chance, Pusteblumen hinzuzufügen.
   * **Pflanzenstatus überprüfen**:
     + CheckPlantStatus(): Reduziert Wasser- und Nährstofflevels basierend auf Wetterbedingungen und überprüft den Status der Pflanze.
   * **Sonstiges**:
     + ShowChecklist(): Zeigt die erreichten Erfolge des Benutzers an.
     + Shop(): Erlaubt dem Benutzer, im Shop Boosts für Wasser, Licht und Dünger zu kaufen, basierend auf der Anzahl der Pusteblumen.

**Fazit**

Der Code implementiert eine virtuelle Pflanzenpflege-App, die dem Benutzer ermöglicht, seine Pflanze durch Gießen, Licht geben, Düngen, Unkraut entfernen und Wachstum zu pflegen. Er beinhaltet Zufallsereignisse wie Wetteränderungen und Unkrautwachstum, die den Zustand der Pflanze beeinflussen. Die Erfolge und das Shop-System bieten zusätzliche Anreize für den Benutzer.