

# Twisted Worlds Wiki

## Klassen:

CameraPositions  
CanvasRefresh  
ChangePlace  
ChapterOneDialogs  
ChapterOneSprites  
Dialog  
InteractableObject  
InventoryManager  
ItemCollectable  
ItemDescriptions  
ItemObject  
ItemPool  
ItemSprites  
MainMenu  
Notes  
PersonClickEvent  
QuickTravelMap  
QuickTravelPoint  
ScrollIntro  
Storyflow  
Talkbox\_fit\_to\_situation

## Enumerations:

CameraPos  
RiddleCameraPos  
Item  
Character  
GameState

## Beschreibung:

### **CameraPositions**

In dieser Klasse werden alle verfügbaren Kamerapositionen gespeichert. Auf jeder dieser Positionen kann eine Szenerie im Unity Editor eingefügt werden. Die Kamerapositionen sind als CameraPos-Wert außerhalb der Klasse ansprechbar und sind wie folgt benannt: <x-position>\_<y-position>

x → y ↓	ZERO	ONE	TWO	THREE	FOUR	FIVE
ZERO	zero_zero	one_zero	two_zero	three_zero	four_zero	five_zero

ONE	zero_one	one_one	two_one	three_one	four_one	five_one
-----	----------	---------	---------	-----------	----------	----------

#### Variablen:

private static:

Vector3 POSITION\_ZERO\_ZERO  
 Vector3 POSITION\_ONE\_ZERO  
 Vector3 POSITION\_TWO\_ZERO  
 Vector3 POSITION\_THREE\_ZERO  
 Vector3 POSITION\_FOUR\_ZERO  
 Vector3 POSITION\_FIVE\_ZERO

Vector3 POSITION\_ZERO\_ONE  
 Vector3 POSITION\_ONE\_ONE  
 Vector3 POSITION\_TWO\_ONE  
 Vector3 POSITION\_THREE\_ONE  
 Vector3 POSITION\_FOUR\_ONE  
 Vector3 POSITION\_FIVE\_ONE

#### Methoden:

private:

void start()

Initialisiert alle Variablen dieser Klasse

void Update()

Bewegt die Kamera zu neuer Position falls gesetzt. Blendet schwarzen Bildschirm ein oder aus falls gesetzt.

public:

void changePosition(CameraPos pos)

Setzt die aktuelle Kameraposition auf übergebene position pos und startet Bewegung der Kamera zu neuer Position. Diagonale Kamera Positionswechsel sind nicht vorgesehen und müssen vermieden werden.

void goToRiddle(RiddleCameraPos pos)

Setzt Kameraposition auf RiddleCameraPos pos und startet Einblenden von schwarzem Bildschirm.

void returnFromRiddle(CameraPos pos)

Setzt Kameraposition auf CameraPos pos und startet Ausblenden von schwarzem Bildschirm.

void fadeInBlender()

Startet Einblenden von schwarzem Bildschirm.

void fadeOutBlender()

Startet Ausblenden von schwarzem Bildschirm.

## **CanvasRefresh**

Meldet Methode des Canvas als Listener des SceneLoaded Events an und aktualisiert den Canvas als Listener bei Objekten in der aktuellen Scene. Sollte bereits ein Canvas existieren wird der neu erzeugte Canvas zerstört.

### **Methoden:**

private:

void Awake()

Zerstört diesen Canvas falls bereits ein Canvas existiert.

void OnEnable()

Methode OnSceneLoaded wird als Listener zum SceneLoaded Event hinzugefügt.

void OnDisable()

Methode OnSceneLoaded wird als Listener vom SceneLoaded Event entfernt.

void OnSceneLoaded(Scene scene, LoadSceneMode mode)

Meldet Canvas als Listener bei relevanten Objekten in aktueller Scene an.

## **ChangePlace**

Component für GameObject, welches zum Wechsel der Szenerie dient. Die im targetPlace gesetzte Kameraposition ist die Ziel-Szenerie. Wird targetPlace auf riddle gesetzt ist die Ziel-Szenerie ein Rätsel und die im targetRiddle gesetzte Rätselposition wird zur Ziel-Szenerie. Nach Beendigung des Rätsels dient die Kameraposition afterRiddle als Ziel-Szenerie. Wird targetPlace nicht auf riddle gesetzt können die anderen Variablen ignoriert werden.

### **Variablen:**

public:

CameraPos targetPlace

RiddleCameraPos targetRiddle

CameraPos afterRiddle

### **Methoden:**

public:

void goTo()

Startet Szenerie-Wechsel basierend auf gesetzten Variablen

## **ChapterOneDialogs**

Enthält alle Dialoge und Texte für das erste Kapitel.

### **Variablen:**

Dialoge mit NPC's:

```
public static List<Dialog> NPC_1
```

Einfache Texte, die den Spieler z.B. informieren, vor Entscheidungen stellen und den Verlauf der Story erklären können:

```
public static List<string> RIDDLE_1_INTRO
```

Einzelner Text, z.B. zum Beschriften von Buttons:

```
public static string BUTTON_1_TEXT
```

Methoden:

private:

```
void start()
```

Initialisiert Variablen.

### **ChapterOneSprites**

Enthält alle Sprites für das erste Kapitel, die als Variable verfügbar sein müssen.

Variablen:

```
public static Sprite PIC_1
```

```
...
```

Methoden:

private:

```
void start()
```

Lädt Sprites. Die Quelldatei muss im Ordner Assets/Resources hinterlegt sein.

### **Dialog**

Basisklasse. Repräsentiert einen einzelnen Dialog mit einem Bild des Dialog führenden und dessen Text.

Variablen:

private:

```
Sprite character
```

```
string text
```

Methoden:

public:

```
Dialog(Sprite c, string txt)
```

Konstruktor.

Sprite getCharacter()  
Übergibt Sprite character

string getText()  
Übergibt string text

### **InteractableObject**

Component für ein Objekt in der Welt welches bei Interaktion stets den gleichen Dialog auslöst.

#### **Variablen:**

private:  
List<Dialog> dialog  
    Dialog zum übergeben an die Klasse Storyflow  
Storyflow storyflow  
    Component Storyflow

public:  
Sprite objectToInteract  
    Sprite zur Anzeige im Dialogfenster.  
string textToDisplay  
    Text zur Anzeige im Dialogfenster.

#### **Methoden:**

private:  
void Start ()  
    Initialisiert die Variable dialog mit objectToInteract und textToDisplay

public:  
void clickOnObject()  
    Leitet Dialog dialog weiter zur Klasse Storyflow und startet so einen Dialog

### **InventoryManager**

Verwaltet Items im Inventar, Anzeige der Items mit Beschreibung und Ein- und Ausblenden des Inventars.

#### **Variablen:**

private:  
Vector3 hidePos  
    Inventar ist auf dieser Position nicht sichtbar.  
Vector3 showPos

Inventar ist auf dieser Position sichtbar.  
Color transparent  
Farbe mit Alpha-Wert 0.  
Color non\_transparent  
Farbe mit Alpha-Wert 255.  
List<ItemObject> items  
Liste aller vom Spieler getragenen Items.  
List<Image> itemDisplay  
Liste der 10 Image-Components zur Darstellung der Items.  
Text itemDescription  
UI Text-GameObject zur Darstellung der Item Beschreibung.  
int currentPage  
Derzeit angezeigte Seite der Items.

#### Methoden:

private:  
void Start ()  
Initialisiert Variablen.

void loadItemPage()  
Füllt die Image-Components des ItemDisplay mit den 10 anzuzeigenden Items.

public:  
void showInventory()  
Setzt die Position des Inventars auf showPos und passt die Anzeige der Items an.

void hideInventory()  
Setzt die Position des Inventars auf hidePos.

void addItem(ItemObject item)  
Fügt ein ItemObject zu items hinzu und passt die Anzeige der Items an.

void removeItem(ItemObject item)  
Entfernt ein ItemObject aus items und passt die Anzeige der Items an.

void nextPage()  
Zeigt bis zu 10 Nachfolgenden Items an.

void previousPage()  
Zeigt die Vorherigen 10 Items an.

void showDescription(int number)  
Zeigt den Text des Items in itemDescription an, welches sich auf der Position number des itemDisplays befindet. number = [0,...,9]

void hideDescription()  
Setzt Text der itemDescription auf null.

### **ItemCollectable**

Component für GameObject welches der Spieler aufsammeln können soll. Das GameObject sollte eine Collider2D Component und eine SpriteRenderer Component haben.

#### Variablen:

public:

Item item

Ein item aus der Enumeration Item.

bool isCollectable

Gibt an, ob das GameObject von Beginn an aufgesammelt werden kann.

#### Methoden:

private:

void Start()

Deaktiviert gegebenenfalls den Collider des GameObject.

public:

void collect()

Item wird zum Inventar hinzugefügt, Collider und SpriteRenderer werden deaktiviert und die Methode ItemCollected() der Klasse Soryflow wird aufgerufen.

void setCollectable()

Aktiviert den Collider des GameObject.

### **ItemDescriptions**

Enthält die Beschreibungen für alle Items in Textform.

#### Variablen:

public:

static string DESCRIPTION\_ITEM\_1

...

### **ItemObject**

Basisklasse. Repräsentiert ein vollständiges Item.

#### Variablen:

private:

Item thisItem

int id

Sprite image  
string description

Methoden:

public:  
ItemObject(Item i, int itemID, Sprite picture, string text)  
Konstruktor.

Item getItem()  
Übergibt Item thisItem.

int getID()  
Übergibt id.

Sprite getImage()  
Übergibt Sprite image.

string getDescription()  
Übergibt description.

**ItemPool**

Diese Klasse initialisiert und speichert alle im Spiel verfügbaren Items. Sie dient als Datenbank und übergibt bei Bedarf einzelne ItemObjects, jedoch können weder die Liste der Items, noch einzelne Items nachträglich bearbeitet werden.

Variablen:

private:  
  
int currentID  
List<ItemObject> items

Methoden:

private:  
void start()  
Initialisiert ItemObjects und speichert diese in der Liste items. Jedes Item erhält eine eindeutige ID.

public:  
ItemObject getItem(int id)  
Übergibt ItemObject mit ID id falls vorhanden, ansonsten null.

ItemObject getItem(Item item)  
Übergibt erstes ItemObject mit Item item aus der Liste der Items.



## **ItemSprites**

Diese Klasse speichert alle Sprites für Items.

### Variablen:

```
public:  
static Sprite ITEM_1_SPRITE  
...
```

### Methoden:

```
private:  
void Awake()  
Lädt alle Sprites aus dem Assets/Resources Ordner.
```

## **MainMenu**

Klasse zur Verwaltung des Hauptmenüs

### Methoden:

```
public:  
void NewGame(string newGameLevel)  
Startet neues Spiel.
```

```
void ExitGameBtn()  
Schließt das Programm.
```

```
void ReturnToMainMenu()  
Lädt das Hauptmenü.
```

## **Notes**

Verwaltet das Notizbuch im Inventar

### Variablen:

```
private:  
Vector3 showPos  
Vector3 hidePos
```

### Methoden:

```
private:  
void Start ()  
Initialisiert Variablen
```

public:  
void showBook()  
Setzt das Notizbuch auf sichtbare Position

void hideBook()  
Setzt das Notizbuch auf nicht sichtbare Position

### **PersonClickEvent**

Component für GameObject, welches als NPC agiert und ansprechbar sein soll. Dabei identifiziert die Variable character den NPC. Das GameObject muss die Methode clickOnPerson über einen EventTrigger aufrufen.

#### Variablen:

public:  
Character character

private:  
GameObject data

#### Methoden:

private:  
void start()  
Initialisiert data.

public:  
void clickOnPerson()  
Ansprechen des Characters wird an Storyflow weitergeleitet.

### **QuickTravelMap**

Zuständig für das Anzeigen oder Verbergen der Schnellreise Karte.

Variablen:  
Vector3 showPos  
Vector3 hidePos

#### Methoden:

private:  
void Start()  
Initialisiert Variablen.

public:

void showMap()

Setzt Position der Karte auf showPos.

void hideMap()

Setzt Position der Karte auf hidePos.

### **QuickTravelPoint**

Punkt auf der Schnellreise Karte welcher einen Ort repräsentiert.

#### Variablen:

public:

string scene

Name der Unity Scene welche bei Bedarf geladen werden soll.

string ingameName

Name des Ortes, der im Spiel als Information angezeigt wird.

#### Methoden:

public:

void travel()

Lädt Scene scene und schließt das Inventar.

void showDetails()

Zeigt Information zum Reiseziel an.

hideDetails()

Entfernt Information zum Reiseziel.

### **ScrollIntro**

Zuständig für das Abspielen des Intros am Anfang

#### Variablen:

public:

float speed

Geschwindigkeit in der das Intro abgespielt werden soll. Wert wird bei Unity eingegeben

private bool move = true

Gibt an, ob sich das Intro derzeit bewegt.

Vector3 startPosition

Startposition des Intro Textes

### Methoden:

IEnumerator LoadLevel()

Intro läuft 20 Sekunden. Anschließend endet das Intro und die Kamera wird ausgeschaltet.  
Bleibt die Kamera weiterhin an, so würde diese die MainCamera überdecken.

private:

void Start()

Startet das Intro

void Update()

Ist verantwortlich dafür, dass sich das Intro bewegt.

public:

void repeatIntro()

Startet Intro erneut.

void exitIntro()

Zerstört alle zum Intro gehörigen GameObjects und startet den ersten Dialog des Fuchses.

### **Storyflow**

Verwaltet Dialoge, Ereignisse, den GameState des Spielers und alle Storyabhängigen Elemente im Spiel.

### Variablen:

public:

enum State

Fortschritt des Spielers wird als State gespeichert

private:

State currentState

Derzeitiger Fortschritt

GameState gState

Derzeitiger Zustand des Spielers

Character currentCharacter

Character, mit dem derzeit eine Konversation geführt wird

int dialogCount

Anzahl aller Dialoge in einer Konversation

int currentDialog

Derzeitiger Dialog, der angezeigt werden soll

GameObject dialogBox;

GameObject welches die Dialoge anzeigt

GameObject leftChoice

Linker Button, der bei einer Entscheidung als mögliche Antwort angezeigt wird  
 GameObject rightChoice  
 Rechter Button, der bei einer Entscheidung als mögliche Antwort angezeigt wird  
 Talkbox\_fit\_to\_situation talkbox  
 Component der dialogbox, steuert Anzeige der Dialoge und position der dialogBox  
 Text storybox  
 Component welche Texte des Erzähler darstellt  
 AudioSource soundDestroy  
 Sound zur Zerstörung der Schatulle  
 bool DeerSonFirstDialog  
 Gibt an, ob der erste oder ein Folgedialog des Characters angezeigt werden soll  
 bool DeerDaughterFirstDialog  
 Gibt an, ob der erste oder ein Folgedialog des Characters angezeigt werden soll  
 bool FoxDaughterFirstDialog  
 Gibt an, ob der erste oder ein Folgedialog des Characters angezeigt werden soll  
 bool StorkFirstDialog  
 Gibt an, ob der erste oder ein Folgedialog des Characters angezeigt werden soll  
 bool ChipmunkFirstDialog  
 Gibt an, ob der erste oder ein Folgedialog des Characters angezeigt werden soll

#### Methoden:

private:

void Start ()

Initialisiert Variablen.

void toDialogue(List<Dialog> dialog)

Steuert Anzeige eines Dialoges auf Basis der übergebenen Liste dialog und dem Fortschritt der Konversation.

void tellStory(List<string> story)

Steuert Anzeige des Erzählertextes in der storybox.

public:

void continueTalk()

Wird aufgerufen, wenn bereits eine Konversation läuft und diese fortgeführt werden soll.

void talkTo(Character character)

Basierend auf Fortschritt des Spielers werden unterschiedliche Konversationen geführt oder nach Ende einer Konversation Anpassungen an den weiteren Storyverlauf vorgenommen.

void interactWithObject(List<Dialog> dialog)

Startet einen Dialog über ein Objekt.

void finishedFadeln()

Der Bildschirm ist nun vollkommen schwarz und Texte des Erzählers können angezeigt werden.

void finishedFadeOut()

Der Bildschirm ist nun nicht mehr schwarz.

void chooseLeft()

Spieler hat sich in einer Entscheidung für den linken Button entschieden.

void chooseRight()

Spieler hat sich in einer Entscheidung für den rechten Button entschieden.

void ItemCollected(Item item)

Wird aufgerufen, wenn der Spieler ein Item erhält.

void continueStory()

Der Bildschirm ist schwarz und der Text des Erzählers wird angezeigt. Basierend auf dem Fortschritt des Dialogs und dem currentState wird die Story fortgesetzt.

void nextState()

Wechselt den currentSate vom aktuellen State in den Nachfolgenden State.

GameState getGameState()

Übergibt aktuellen GameState.

### **Talkbox\_fit\_to\_situation**

Component für GameObject DialogBox. Steuert das Ein- und Ausblenden der gesamten DialogBox inklusive Anpassung der Sprites und Texte.

#### **Variablen:**

public:

int speed

Geschwindigkeit, mit welcher die Dialogbox ein- bzw. ausgeblendet wird.

private:

Color transparent

Farbe mit Alpha-Wert 0.

Color non\_transparent

Farbe mit Alpha-Wert 255.

GameObject diaWindow

Das GameObject DialogWindow.

GameObject leftImg

GameObject für die bildliche Anzeige des Dialogführenden.

GameObject txtbox

GameObject für die Anzeige des Dialogtextes.

Vector3 hidePos

Position, in der die Dialogbox nicht sichtbar ist.

Vector3 showPos

Position, in der die Dialogbox sichtbar am unteren Rand des Bildschirms ist.

bool showBox

Gibt an, ob die Dialogbox derzeit eingeblendet wird und somit in Bewegung ist.  
bool hideBox

Gibt an, ob die Dialogbox derzeit ausgeblendet wird und somit in Bewegung ist.

#### Methoden:

private:

void Start ()

Initialisierung der Variablen

void Update()

Bewegt die Dialogbox in Richtung hidePos oder showPos falls gewünscht.

void activate(string m\_text)

Setzt m\_text als Dialogtext ein und aktiviert die Button-Komponente des DialogWindow um clicks abzufangen sowie das Einblenden der Dialogbox.

public:

void show(Sprite leftImage, string text)

Setzt leftImage als Bild des Dialogführenden und gibt text an die Methode activate() weiter.

void update(Sprite leftImage, string text)

Ersetzt Text und Bild durch neue Übergebene Variablen.

void hide()

Deaktiviert die Button-Komponente des DialogWindow und aktiviert das Ausblenden der Dialogbox.