# Projektdokumentation

Projekt

Infinity Spiegel

[Projektdokumentation 1](#_Toc168388839)

[Zusammenfassung 3](#_Toc168388840)

[Infinity-Spiegel 3](#_Toc168388841)

[Problemstellung 3](#_Toc168388842)

[Lösungsweg 3](#_Toc168388843)

[Themenwahl 3](#_Toc168388844)

[Erste Schritte 3](#_Toc168388845)

[Display 4](#_Toc168388846)

[Wetterdaten 4](#_Toc168388847)

[Displayanzeige 6](#_Toc168388848)

[GPS-Modul 6](#_Toc168388849)

[LED-Lichtstreifen 6](#_Toc168388850)

# Zusammenfassung

## Infinity-Spiegel

Ein Infinity-Spiegel ist ein Spiegel, der die Illusion erzeugt in einen unendlich tiefen Raum zu schauen. Dieser Effekt wirkt sehr beeindruckend, sodass wir uns endschieden haben, ihn für unser Projekt zu nutzen. Der Spiegel wird mit steuerbaren LED-Lichtstreifen und einem Display ausgestattet, sodass wir unsere Software an diesen Geräten anwenden können. Das Display soll die Funktion besitzen, aktuelle Wetterdaten sowie den aktuellen Standort anzuzeigen. Die Wetterdaten enthalten ein Icon, den allgemeinen Wetterstatus, aktuelle Temperatur, Windrichtung und auch die Windgeschwindigkeit. Diese Daten und der aktuelle Standort sollen sich in einem definierten Zeitraum aktualisieren. Der LED-Lichtstreifen ist für den Infinity-Spiegel essenziel, da er zur Erzeugung der „Unendlichkeit Illusion“ beisteuert. Für diesen Effekt muss der Lichtstreifen das Licht zwischen den beiden Spiegeln erzeugen. Da das Licht den Effekt beeinflusst, sind sehr anschauliche „Spielereien“ mit dem Ansteuern des Lichtstreifens möglich.

# Problemstellung

Die Problemstellung dieses Projekts besteht darin, ein Display für den Infinity-Spiegel so zu programmieren, dass dieser periodisch Wetterdaten aktualisiert und sie auf eine ansprechende und leicht lesbare Weise darstellt. Des Weiteren muss der LED-Lichtstreifen so abgestimmt sein das der Infinity-Effekt hervorragend zur Geltung kommt. Da es mit einem Belichtungs-Modus zu monoton wird, werden hier mehrere Modi angestrebt. Zudem gibt es eine weitere, eigentlich nicht für das OOP-Projekt vorgesehene, weiter Aufgabe: Den Infinity-Spiegel zu bauen.

# Lösungsweg

## Themenwahl

Das Thema wurde aufgrund der zwei unterschiedlichen Anforderungen gewählt. Der erste Aspekt betrifft die Hardwareansteuerung des LED-Lichtstreifens und der Displayausgabe. Der Zweite Aspekt, ist die softwareabhängige Abfrage der Wetterdaten. Dies schien ein perfektes Projekt für objektorientierte Programmierung zu sein.

## Erste Schritte

Als Erstes stellte sich die Frage, wo man einen Infinity-Spiegel herbekommt. Da wir schnell bemerkten, dass diese sehr teuer sind, entschlossen wir uns den Spiegel selbst zu bauen. Somit fuhren wir in den Baumarkt kauften alle benötigten Teile und bauten unseren eigenen Infinity-Spiegel. Jetzt konnten wir uns ganz auf die Software und die beteiligte Hardware konzentrieren.

## Display

### Wetterdaten

Für die Wetterdaten wird eine kostenlose API verwendet. Diese gibt die Daten der nächstgelegenen Wetterstation, die dem Server zur Verfügung steht, aus. Um die Wetterdaten zu erlangen, muss man eine get-request an die API schicken. Für diese get-request benötigt man jedoch noch die Breiten- und Längengrade um den Standortfestzulegen. Um diese zu erhalten, wird ein GPS-Modul verwendet. Auf die Implementierung des GPS-Modul und das Programm für die Abfrage der Breiten- und Längengrade wird später eingegangen.

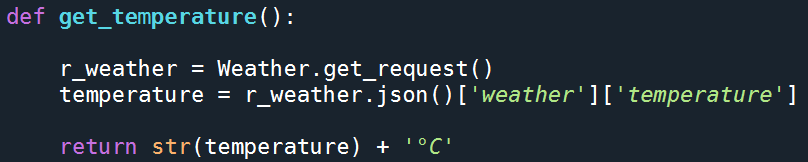
Für die Wetterabfrage wurde die Klasse Weather erstellt. Diese enthält nur Funktionen und keine Attribute. Somit dient sie nur als Funktionssammlung und muss nicht initialisiert werden.

Um eine Request an die API zu senden wird die Funktion „get\_request“ verwendet.

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Die Funktion fragt die Koordinaten über eine weiter Funktion ab. Diese werden dann in einen String konvertiert, um sie in die request.get-Funktion einfügen zu können. Die request.get-Funktion gibt eine Art Dictionary mit dem Datentyp Json aus. Diese Json Datei wird dann über return zurückgegeben.

Für die Temperatur wir die Funktion get\_temperature verwendet.

Diese Funktion führt die get\_request-Funktion aus. Wandelt dann die Json-Datei in ein Dictionary um und sucht nach dem Value für den Key „temperature“. Dieser wird dann mit return zurückgegeben, davor wird er aber noch in einen String konvertiert und ein „°C“ hinzugefügt.

Für die Winddaten wird die Funktion get\_wind verwendet. Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Diese Funktion führt auch die get\_request-Funktion aus. Wandelt diese in ein Dictionary um und sucht nach dem Value vom Key „wind\_speed\_30“ und den Key „wind\_gust\_direction\_30“. Daraus erhalten wir die Windgeschwindigkeit in km/h der letzten 30 Minuten und die Windrichtung in Grad in den letzten 30 Minuten. Damit die Windrichtung nicht in Grad angezeigt wird, sondern die ungefähre Himmelsrichtung, durchläuft die Windrichtungsinformation in Grad noch die If-Schleifen. Diese Werten aus in welchem Intervall die Windrichtung liegt und speichert die passende Himmelsrichtung in der Variable wind\_direction ab. Diese wird mit return zurückgegeben zusammen mit dem String „Wind“, desweiterem wird die Windgeschwindigkeit auch mit zurückgegeben.

Für den Status wird die Funktion get\_status verwendet.

Ein Bild, das Text, Schrift, Screenshot enthält.

Automatisch generierte Beschreibung

Diese Funktion sucht nach der Request und der Konvertierung nach dem Value vom Key „Icon“. Gibt dieses dann aus.

Für das passende Icon wird die Funktion get\_icon verwendet.

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Das Funktionsprinzip ist dasselbe wie bei der get\_status-Funktion. Nur muss diese noch das Value in den Path einsetzen, so dass dieser übergeben werden kann. Mit öffnen der Datei über den Path wird das passende Icon mit dem Datentyp JPG aufgerufen.

Die Icons wurden in dem weather\_icons-Ordner abgespeichert. Zudem wurden noch passende Dateinamen gewählt, die mit dem Status übereinstimmen, sodass es die Programmierung erleichterte.

Ein Bild, das Screenshot, Text, Design enthält.

Automatisch generierte Beschreibung

<https://www.iconfinder.com/search?q=weather>

Fazit Weather-Klasse:

Die Klasse Weather kann somit für alle Anwendungen die Wetterdaten benötigt verwendet werden. Außerdem kann sie noch erweitert werden für weitere Wetterdaten, wie zum Beispiel den aktuellen Luftdruck.

### Displayanzeige

Für die Displaydarstellung wurde eine bereits vorhandene Bibliothek verwendet namens Pygame. Diese ermöglicht Grafiken anzuzeigen und zu animieren. Diese wird in das Hauptprogramm importiert so wie unsere Weather-Klasse. Danach muss die Klasse pygame.init() einmal initialisiert werden, zugleich teilen wir der Weather-Klasse eine neuen Platzhalter zu.

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Danach werden die Skalierungen für das Icon festgelegt sowie den Skalierungsfaktor für die Animation. Als nächstes fragen wir den passenden Icon-Path zu den aktuellen Wetterdaten ab, anhand unserer get\_icon-Funktion ab. Nun werden Text-Objekte eingefügt, den passenden Text laden wir über unsere Funktionen aus der Weather-Klasse. Als letztes legt man noch die Größe des Programmfensters fest, hier verwenden wir den Vollbildschirm.

Ein Bild, das Text, Screenshot, Schrift, Software enthält.

Automatisch generierte Beschreibung

Im Hauptprogramm gibt es eine Hauptschleife und eine Nebenschleife. Das Programm läuft eine festgelegte Zeit in der Hauptschleife, wenn diese unterbrochen wird, werden alle Daten anhand der Funktionen in der Weater-Klasse aktualisiert. Ein Bild, das Text, Screenshot, Schrift, Software enthält.

Automatisch generierte BeschreibungEin Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

## GPS-Modul

## LED-Lichtstreifen