Übung 02

Game Programming SS 2024

Dr. Martin Kocur & Axel Bauer

UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA

Aufgabe: Nutzereingaben abfangen (Wiederholung)

- Erstellen Sie ein Skript namens *GameManager*
- Fangen Sie eine Nutzereingabe ab:
  - Wird "a" auf der Tastatur gedrückt, soll "Attacke" auf der Konsole ausgegeben werden
  - Wird "v" auf der Tastatur gedrückt, soll "Verteidigung" auf der Konsole ausgegeben werden
  - Kommentieren Sie diesen Code im Anschluss aus.
- Wird "Space" auf der Tastatur gedrückt, soll 1 Lebenspunkt des Enemies abgezogen werden. Wird "w" auf der Tastatur gedrückt, soll 1 Lebenspunkt des Spielercharakters abgezogen werden. Mit "h" kann sich der Spielercharakter heilen (Erhöhen der Lebenspunkte um 1)
  - *Es werden keine 3D Modelle und Visuals benötigt. Eine Repräsentation der Lebenspunkte über eine Ganzzahl ist ausreichend.*
- Bonus: Sinken die Lebenspunkte auf 0, so soll das jeweilige GameObject verschwinden/zerstört werden.

Scopes


Source: https://fi-static.luehrsen-heinrich.de/wp-content/uploads/2016/06/442c7f9e05cdc69037b8ec81db4dc9fe.jpg

Scopes

```
int myNumber = 1;
Debug.Log(myNumber); // -> 1

{
    Debug.Log(myNumber); // works as well!
    myNumber = myNumber + 1; // -> 2
}

Debug.Log(myNumber); // -> 2 recognises the change inside the scope
```

Funktioniert: Von innen nach außen auf Variablen zugreifen

```
{
    string myString = "hello"; // I'm hiding inside the scope
    Debug.Log(myString); // works perfectly fine
}

Debug.Log(myString); // doesn't work!
```

Fehler: Auf in dem Scope erstellte Variablen zugreifen

Scopes

Achtet auch immer darauf, den gleichen Variablennamen nicht doppelt zu verwenden!

```
float myFloat = 1.0f;
{
    float myFloat = 2.0f;

    // doing something with myFloat
}
```

Fehler: Der Compiler und auch wir kennen uns nicht mehr aus, welcher Wert hergenommen werden soll

Scopes – Anwendungsbeispiele (1)

IF-Bedingungen

```
int myInt = 0;

if (myInt == 0) // any condition
{
    myInt = myInt + 1;
}
```

(FOR-) Schleifen

```
string funnyJokeReaction = "";

for (int i = 0; i < 10; i++)
{
    funnyJokeReaction = funnyJokeReaction + "ha";
}

Debug.Log(funnyJokeReaction); // hahahahahahahahahaha
```

Scopes – Anwendungsbeispiele (2)

Methoden / Funktionen !

```
Vector3 myVector = new Vector3(0,0,0);

// Start is called before the first frame update
Event function
void Start()
{
    Debug.Log(myVector);
}

// Update is called once per frame
Event function
void Update()
{
    myVector.x = myVector.x + 1;
}
```

Vektoren?

Kann mehrere Zahlenwerte speichern!

Vector2 -> 2 Werte
Vector3 -> 3 Werte

Sehr hilfreich bei räumlichen Daten (Koordinaten!)

Vector2 -> (x, y)
Vector3 -> (x, y, z)

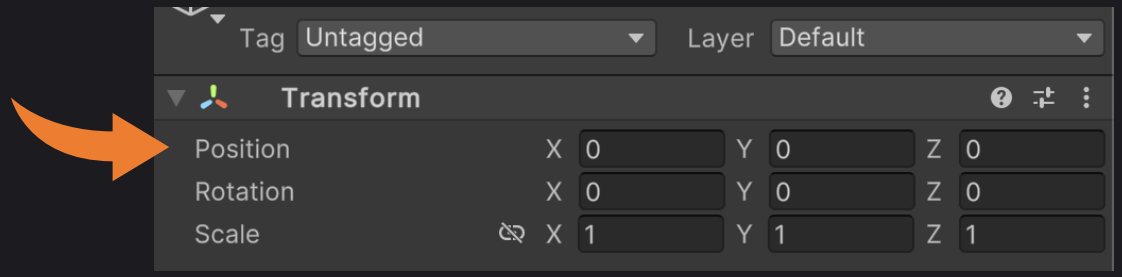
Mit Vektoren kann man einfach rechnen:

```
Vector2 vec1 = new Vector2(1, 2);
Vector2 vec2 = new Vector2(3, 4);

// vec1 + vec2 = (1+3, 2+4) = (4,6)
```

Oder einzelne Bestandteile von ihnen hernehmen:

```
myVector.x = myVector.x + 1;

myVector[0] = myVector[0] + 1; // same as above
```

Kennen wir schon aus Unity!



Aufgabe: Recherchiert, wie man ein Objekt bewegen kann

- Erstellen Sie ein Skript namens *Player*
- Fangen Sie eine Nutzereingabe ab:
  - Wird "w" auf der Tastatur gedrückt, soll sich ein Objekt (z. B. Cube) vorwärts

- Bonus: Ergänzen Sie das Skript für den restlichen räumlichen Achsen (rückwärts, links, rechts, oben?, unten?)

- Hilfestellung: Erinnert euch daran, wie wir Input in Unity erfasst und mit Variablen addiert/subtrahiert haben. Auch das neue Wissen über Vektoren (-> Position) ist hilfreich!

Dokumentation

- Input.GetKey() - solange die Taste gedrückt ist
- Input.GetKeyDown() - pro EIN MAL Tastendruck
- KeyCode

- Vector3 - Vector3.Forward, Vector3.right, … etc
- Transform - Beinhaltet die Variablen (z. B. position) und Methoden zum Bewegen (z. B. Translate()) eines GameObjects

Scopes (Teil 2)

Private vs. Public

Source: https://fh-static.luehrsen-heinrich.de/wp-content/uploads/2016/06/442c7f9e05cdc6903708ec8cb4c9fe.jpg

Scopes: Private vs. Public

= Sichtbarkeit außerhalb der Klasse (Datei)

Private

- Keinen Unterschied zu dem, wie wir bis jetzt Variablen verwendet haben
- Geeignet, für Variablen, die nur in dieser Datei gebraucht werden
- Sollte bei jeder Variable die nicht "public" ist gesetzt werden

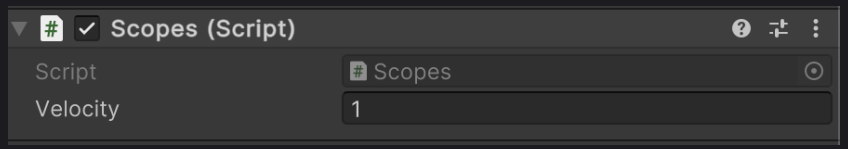
```
private float size = 0;
```

Public

- Sollte so selten als möglich / nötig verwendet werden
- Variable ist auch außerhalb der Klasse (Datei) sichtbar

```
public float velocity = 1; // visible!
```

- Kann nun auch in Unity verändert werden!



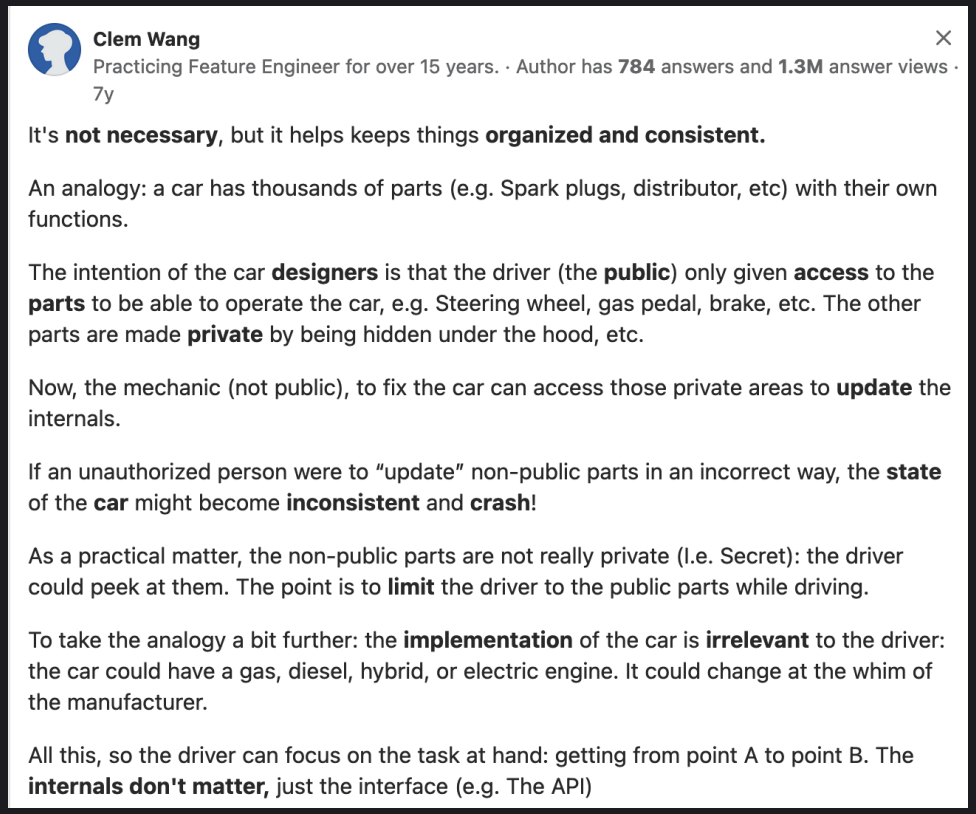
Scopes: Private vs. Public

Warum?

- Bewusste Einteilung, was sichtbar ist und was nicht (= Vorteil, bessere Übersicht/Strukturierung für dich!)
- Macht es einfacher anderen zu zeigen, welche Variablen/Methoden (nämlich die public) verwendet werden sollen, die nicht wissen müssen, wie der Code dahinter funktioniert
- Wenn alle die vorgegebenen Variablen / Funktionen verwenden, kannst du den Code dahinter in Zukunft ohne Probleme anpassen, ohne dass andere (Programmbestandteile) angepasst werden müssen!

Scopes: Private vs. Public

Analogie:



Clem Wang
Practicing Feature Engineer for over 15 years. · Author has **784** answers and **1.3M** answer views · 7y

It's **not necessary**, but it helps keeps things **organized and consistent.**

An analogy: a car has thousands of parts (e.g. Spark plugs, distributor, etc) with their own functions.

The intention of the car **designers** is that the driver (the **public**) only given **access** to the **parts** to be able to operate the car, e.g. Steering wheel, gas pedal, brake, etc. The other parts are made **private** by being hidden under the hood, etc.

Now, the mechanic (not public), to fix the car can access those private areas to **update** the internals.

If an unauthorized person were to "update" non-public parts in an incorrect way, the **state** of the **car** might become **inconsistent** and **crash**!

As a practical matter, the non-public parts are not really private (I.e. Secret): the driver could peek at them. The point is to **limit** the driver to the public parts while driving.

To take the analogy a bit further: the **implementation** of the car is **irrelevant** to the driver: the car could have a gas, diesel, hybrid, or electric engine. It could change at the whim of the manufacturer.

All this, so the driver can focus on the task at hand: getting from point A to point B. The **internals don't matter,** just the interface (e.g. The API)

https://www.quora.com/Why-is-scope-public-private-protected-etc-necessary-or-important-in-programming

Scopes: Private vs. Public

Fachbegriffe

- Verkapselung:
  - Steuerung von Sichtbarkeit und Zugänglichkeit von anderen Teilen des Programms
- Sicherheit:
  - Verhindert unbefugten Zugriff und verhindert unbeabsichtigte Änderungen (die eventuell das ganze Programm lahm legen könnte
- 3. Abstraktion:
  - Bereitstellung klare und vereinfachte Schnittstelle für die Interaktion mit anderen Teilen des Programms bereitstellen (public)
- 4. Wartbarkeit:
  - Erleichtert die Wartung und das Verständnis der Codebasis.
  - Macht deutlich, wo und wie bestimmte Elemente des Programms verwendet werden sollten.