**Unity3D**

* **eine Entwicklungsumgebung für 2D und 3D Spiele**

Inhaltsverzeichnis

1. Die ersten Schritte 3

1.1 Download und Installation 3

1.2 Foren und Tutorials 3

2. Das Interface 3

3. Scene 3

4. User Interface 4

4.1 Canvas 4

5. GameObjects 4

5.1 Object References 4

6. Monobehaviour 4

7. Nützliches beim Scripten 5

7.1 Ausgabe in der Console (Debugging) 5

7.2 Einfach Spielerdaten Zwischenspeichern: PlayerPrefs 5

7.3 Coroutines 5

7.4 Alternative zum MVC Pattern → AMVCC 5

8. Building 5

9. Datenbankanbindung 5

9.1 Einrichten der Datenbank 5

10. Debugging 6

1. 1. Die ersten Schritte
   1. 1.1 Download und Installation

* <https://unity3d.com/get-unity>

(Personal Edition ist kostenlos verfügbar)

* 1. 1.2 Foren und Tutorials

Unter der oben genannten Website (1.1) lassen sich wichtige Hinweise für die ersten Schritte sowie das weitere Arbeiten mit Unity3D finden.

* **Learn**: Tutorials und die API

(Anfängertutorials unter: Topics → Interface & Essentials;

API unter Documentation: Unity Overview → Unity Basics)

* **Community**: Foren + Q&A's
* beim ersten Öffnen Account erstellen

1. 2. Das Interface

* Control Tools oben links (mit QWERT kann man zwischen diesen Wechseln)

(Q=Szenenansicht bewegen (ist nicht die Camera!); W=Objekt entlang der Achsen bewegen, auch im Inspector möglich, E=Rotation, R=Maße des Objekts ändern)

* rechts davon: Einstellungen für Anker- und Drehpunkte
* oben mittig: Play, Pause und Next um Spiel zu starten etc
* Hierarchy: alle aktiven Objekte in der aktuellen Scene (Doppelklick fokussiert das Objekt)
* Scene View: Aufbau der Scene (per drag&drop)
* Inspector: Components des ausgewählten Objekts (Eigenschaften ändern, Skripte einem Objekt hinzufügen, Variablenwerte ändern, …)
* Project: Inhalte des Asset-Ordners (Skripte, 2D/3D Modelle, Schriftarten, …)

→ Create → C#, JavaScript, Folder, …

1. 3. Scene

File → New Scene

* referenzierbar durch Index oder Name
* neue Scene laden: Application.LoadLevel("mainMenu"); //mainMenu ist hier der Name der Scene

1. 4. User Interface

Create → UI (canvas, panel, button, ...)

* 1. 4.1 Canvas
* all UI Elemente sind children von einem canvas (wenn keiner existiert wird automatisch einer erstellt)
* Render Mode → wo in der Szene? (default: Overlay=über der Szene, automatisch angepasst [Rect Transform nicht editierbar])

<http://docs.unity3d.com/Manual/UISystem.html>

* Project → Create Folder → Fonts hinzufügen

1. 5. GameObjects

* jedes GameObject ist ein leeres GameObject mit hinzugefügten Components (z.B. Transform für Position + Ausmaße, MeshRenderer zur Darstellung, ...)
* auch ein Skript kann eine Component sein und per drag&drop hinzugefügt werden
* Zerstören des zum Script gehörigen GameObjects: Destroy(gameObject,*delay*)
  1. 5.1 Object References

Bsp: dem Script von Objekt B folgendes Feld geben: public ClassA *name1* (und auch public gameObject *name2*); dann der Script Component des Objekt B's per drag&drop das Objekt A anhängen

→ *name1* = *name2*.GetComponent<name1>();

Dies kann allerdings nur in der main thread aufgerufen werden, es ist also sinnvoll das in Start() zu tun.

1. 6. Monobehaviour

* <http://docs.unity3d.com/ScriptReference/MonoBehaviour.html>
* jede Klasse erbt von Monobehaviour
* Start() → wird auf dem ersten Frame eines aktivierten Skripts aufgerufen
* Update() → wird einmal pro Frame aufgerufen (= 60x pro Sekunde)
* Awake() → wird aufgerufen, wenn das zugehörige GameObject initialisiert wird
* Destroy() → entfernt ein GameObject oder ein Asset

wichtig: wenn du gerade beginnst, nutze Debug.Log (siehe 6.1) in der Start-Funktion um zu überprüfen wie oft sie aufgerufen wird.

→ wird sie öfter als einmal aufgerufen, dann stelle sicher, dass das Script, welches Start ethält nur einem Objekt als Component hinzugefügt wurde; da Start einmal PRO Object ausgeführt wird, welches über ein Script mit Start verfügt

1. 7. Nützliches beim Scripten
   1. 7.1 Ausgabe in der Console (Debugging)

Debug.Log (oder auch Print(), wenn von Monobehaviour geerbt wird)

ebenso möglich: Debug.LogWarning, Debug.LogError

* 1. 7.2 Einfach Spielerdaten Zwischenspeichern: PlayerPrefs

<http://docs.unity3d.com/ScriptReference/PlayerPrefs.html>

Bsp: PlayerPrefs.SetString("PlayerName", playerName); //erster String ist der Key

* 1. 7.3 Coroutines
* äußerst praktisch und thread-safe → gute alternative paralleles Verhalten zu erschaffen (läuft eigentlich gar nicht parallel)
* <http://docs.unity3d.com/Manual/Coroutines.html>
  1. 7.4 Alternative zum MVC Pattern → AMVCC

Application-Model-View-Controller-Component

(Application stellt alle Referenzen zur Verfügung; Components sind wiederverwendbare funktionale Einheiten, die weder M noch V noch C angehören)

<http://www.toptal.com/unity-unity3d/unity-with-mvc-how-to-level-up-your-game-development>

1. 8. Building

File → Build Settings, Plattform aussuchen

Wichtig ist, dass alle zum Projekt gehörenden Szenen hinzugefügt sind!

1. 9. Datenbankanbindung
   1. 9.1 Einrichten der Datenbank

* XAMPP, PostgreSQL, phpPgAdmin herunterladen
* phpPgAdmin nach Ordner xampp entpacken
* xampp → php → ext: php\_pdo\_pgsql.dll, php\_pgsql.dll
* apache → conf → extra → httpd-xampp.conf: Einfügen nach Alias:

#Added to integrate phppgadmin

Alias \phppgadmin "C:xampp/phpPgAdmin/"

<Directory "C:xampp/phpPgAdmin/">

AllowOverride AuthConfig

Require local

</Directory>

* php → php.ini: „extension=php\_pdo\_pgsql.dll“ sowie „extension=php\_pgsql.dll“ einkommentieren (; davor entfernen)
* phppgadmin -> conf -> config.inc.php: wie folgt ergänzen:

// Hostname or IP address for server. Use '' for UNIX domain socket.

// use 'localhost' for TCP/IP connection on this computer

$conf['servers'][0]['host'] = 'localhost';

// Specify the path to the database dump utilities for this server.

// You can set these to '' if no dumper is available.

$conf['servers'][0]['pg\_dump\_path'] = 'C:<\\Program Files\\PostgreSQL\\9.1\\bin\\pg\_dump.exe'>;

$conf['servers'][0]['pg\_dumpall\_path'] = 'C:<\\Program Files\\PostgreSQL\\9.1\\bin\\pg\_dumpall.exe'>;

$conf['extra\_login\_security'] = true; → auf false setzen

* pgadmin3 -> DB -> postgres -> Rechtsklick wiederherstellen: postgres.backup
* xampp -> htdocs: den Ordner „unity\_games“ aus den Assets des Projekts und in den Hauptordner (xampp) die andere Datei (iwm\_ulg\_pgconfig) einfügen
* crossdomain.xml in xampp → htdocs hinzufügen

1. 10. Debugging

* NullReferenceException: Object reference not set to an instance of an object

→ prüfe, ob referenziertes Objekt auch wirklich existiert

* You are trying to create a MonoBehaviour using the 'new' keyword. This is not allowed.

→ wenn von MonoBehaviour geerbt wird, kann der new-Befehl nicht genutzt werden. Nutze stattdessen *object*.AddComponent<*Script*>();

→ welchem Objekt füge ich nun das Script hinzu? Die Main Camera ist immer aktiv und bietet sich daher an, z.B. private View view = GameObject.Find("Main Camera").AddComponent<View>();

* Farben von Buttons neu setzen (z.B. die Farbe für deaktivierte Buttons), da die Buttonfarbe vom Datentyp *struct* ist, gibt sie eine Kopie von sich selbst wieder, der man dann die Farbe zuordnen muss

ColorBlock cb = button.colors; //copy

cb.disabledColor = Color.green; //add change to the copy

button.colors = cb; //apply copy back

* deadlocks bei der Nutzung von coroutines () → kein while vor yield nutzen

→ beim Abfragen von den gleichen Feldern innerhalb und außerhalb der Coroutine bzw. Ändern von Werten und die Coroutine wird danach nicht mehr benötigt: StopAllCoroutines(); bzw StopCoroutine(string methodName);

* SecurityException: No valid crossdomain policy available to allow access

→ Edit → Project Settings → Editor (change Host URL to http://localhost/unity\_games)