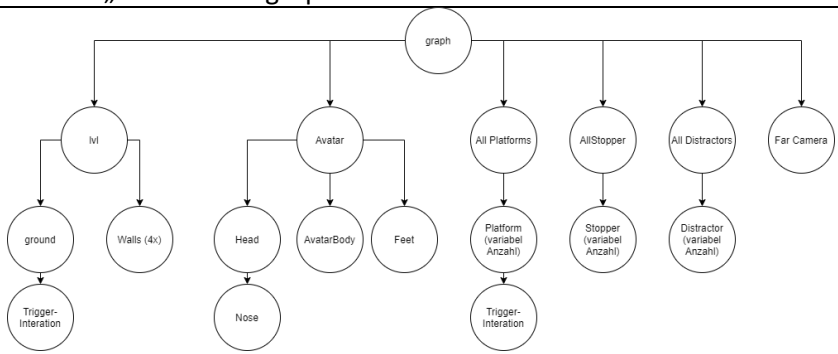
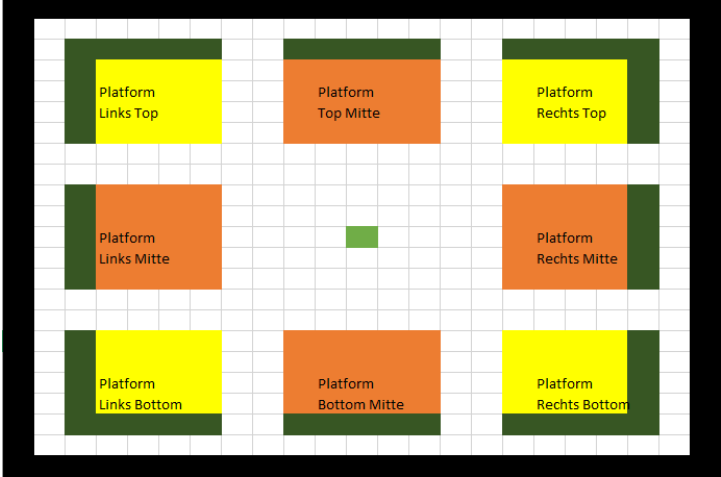


Nr.	Bezeichnung	Inhalt
	Titel	WhackyTowerJump
	Name	Lukas Willmann
	Matrikelnummer	263930
1	Nutzerinteraktion	<ul style="list-style-type: none"> - Steuerung des Avatars mit WASD sowie E und R <ul style="list-style-type: none"> o W S → vorwärts rückwärts o A D → rechts- links- Rotation o E R → hoch- runter- Rotation des Kopfes - Leertaste gedrückt halten: Sprungkraft aufladen - Leertaste loslassen: Sprung mit aufgeladener Sprungkraft ausführen in Blickrichtung - Wechseln der Kameraperspektive mit C - Einsatz der GUI <ul style="list-style-type: none"> o Spiel starten o Spiel pausieren o Spiel neustarten o Anzahl der Plattformen verändern o Wahrscheinlichkeit des „Distractor-Spawns“ hoch oder runter regulieren
2	Objektinteraktion	<ul style="list-style-type: none"> - Kollisionen durch die Physik-Engine und die Rigid-Bodys - Auslösen der Plattform-Trigger <ul style="list-style-type: none"> o Erhöhung des Scores o Hochfahren der nächsten Plattform - Auslösen der „Ground“-Triggers <ul style="list-style-type: none"> o Reduzierung der Lebenspunkte o Runterfahren der Plattformen welche bereits hochgefahren sind - Kollision mit „Distractor“ führt durch „bounceComponentScript“ dazu, dass Spieler in entgegengesetzte Richtung weggeworfen wird
3	Objektanzahl variabel	<ul style="list-style-type: none"> - Plattformen werden zur Laufzeit generiert durch die Eingabe des Spielers, wie viele Plattformen er gern hätte - Anzahl der Stopper für die Plattformen wird nach dem gerichtet, wie viele Plattformen der Spieler wollte und was es für Plattformen sind die zufällig generiert werden - Anzahl der „Distractors“ ist variabel, je nachdem, wie viele Plattformen der Spieler will und wie hoch die Wahrscheinlichkeit eingestellt wird, dass viele oder wenige „Distractors“ spawned werden
4	Szenenhirarchie	 <pre> graph TD graph((graph)) --> lvl((lvl)) graph --> Avatar((Avatar)) graph --> AllPlatforms((All Platforms)) graph --> AllStopper((All Stopper)) graph --> AllDistractors((All Distractors)) graph --> FarCamera((Far Camera)) lvl --> ground((ground)) lvl --> Walls((Walls (4x))) ground --> TriggerInteraction1((Trigger-Interaction)) Avatar --> Head((Head)) Avatar --> AvatarBody((AvatarBody)) Avatar --> Feet((Feet)) Head --> Nose((Nose)) AllPlatforms --> Platform((Plattform (variabel Anzahl))) Platform --> TriggerInteraction2((Trigger-Interaction)) AllStopper --> Stopper((Stopper (variabel Anzahl))) AllDistractors --> Distractor((Distractor (variabel Anzahl))) </pre> <ul style="list-style-type: none"> - Graph ist der überstehende Parent, der abgebildet wird - „lvl“ wird aus einer JSON geladen. Der Graph und die Node „lvl“ wurde im Fudge Editor erstellt

		<ul style="list-style-type: none"> - Avatar beinhaltet alle Komponenten die für den Avatar zuständig sind mit Rotation des Kopfes, Rigidbody, Meshes etc. - „All Plattformen“, „All Stopper“ und all „Distractors“ hätte man nochmal in einen über Node packen können um alles beieinander zu halten, jedoch hätte das meiner Meinung in der Programmierung keinen Mehrwert sondern eher mehr Zeilen gebracht und für die Verhältnisse zueinander macht es keinen Unterschied - „FarCamera“ ist der Node an welchem die Kamera angehängt wird um den Avatar von der Ferne zu sehen. Die Kamera soll dann nicht mit irgendetwas rotieren, deswegen ist sie nicht dem Avatar sondern nur dem Graphen untergeordnet. Dadurch muss dann aber auch immer die Y-Position des Avatars abgeglichen werden
5	Sound	<ul style="list-style-type: none"> - Es existiert eine Hintergrund Musik - Es existiert ein Sprung-Sound - Es existiert ein Sound wenn man die Distractors berührt - Sobald eine der Testen gedrückt wird, die für die Steuerung des Avatars zuständig sind, beginnt die Musik. - Sound wird immer dort gehört wo die Camera ist. Somit muss auch der Listener seinen Platz wechseln, wenn die Kamera irgendwo anders angehängt wird
6	GUI	<ul style="list-style-type: none"> - Highscore (wird gespeichert in Local Storage) - Score der aktuellen Runde - Leben des Avatars - Sprungkraft-Anzeige - Pause Button <ul style="list-style-type: none"> ○ Öffnet ein Pausefenster in welchem nur kurz Pausiert werden kann aber sonst keine Funktion hat - Restart Button <ul style="list-style-type: none"> ○ Öffnet das Optionen Menu, in welchem man wieder die Anzahl der Plattformen und die Wahrscheinlichkeit des „Distractor-Spawnens“ einstellen und danach neu Starten kann
7	Externe Daten	<ul style="list-style-type: none"> - Graph wird aus JSON geladen - Basis Daten des Spiels werden aus einer anderen JSON geladen. Folgende Daten haben Einfluss auf das Spiel: <ul style="list-style-type: none"> ○ „platformArray“ ist eine Nummer, welche den Default-Wert der Plattformen im Spiel angibt. Falls der Spielende keine eingetragenen nachträglichen Werte mit dem Schieber eingibt wird die Anzahl aus dem JSON geladen ○ „disturberProb“ ist eine Nummer die die Wahrscheinlichkeit verstellt, wie oft ein Disturber spawned wird. (gleiches Prinzip wie bei der Plattformmenge)
8	Verhaltensklassen	<ul style="list-style-type: none"> - Avatar <ul style="list-style-type: none"> ○ computeJumpForce() → (in Avatar.ts Zeile 75) <ul style="list-style-type: none"> ▪ berechnet die „jumpForce“ des Avatars ○ hndlJump(_event: KeyboardEvent) → (in Avatar.ts Zeile 88)

		<ul style="list-style-type: none">▪ beeinflusst den Rigidbody nach einem Loslassen der Leertaste des Avatars anhand dessen Blickrichtung, der berechneten „jumpForce“○ recover() → (in Avatar.ts Zeile 100)<ul style="list-style-type: none">▪ Setzt einen Timer, nachdem der Avatar einen Distractor berührt hat, nachdem er sich wieder bewegen kann (wenn der Avatar sich bewegt, während der Distractor ihn wegstoßen will, wird der Effekt nicht so gut ausgeführt. Deswegen muss der Avatar kurzzeitig paralysiert werden)○ isRecovered() → (in Avatar.ts Zeile 104)<ul style="list-style-type: none">▪ setzt ein Attribut des Avatars, dass er sich wieder bewegen darf (wird von recover() ausgelöst)- Platform<ul style="list-style-type: none">○ hndTrigger (_event: fc.EventPhysics) → (in Platform.ts Zeile 45)<ul style="list-style-type: none">▪ löst aus, dass die nächste Plattform hochgeholt wird und erhöht den Score der Gui- Hud<ul style="list-style-type: none">○ start()→ (in Hud.ts Zeile 18)<ul style="list-style-type: none">▪ Funktion welche aus der Vorlesung Prima übernommen und an das Projekt angepasst wurde (vgl. Prima/Hud.ts at master · JirkaDellOro/Prima (github.com))																																																																													
9	Subklassen	<ul style="list-style-type: none">- GameObject<ul style="list-style-type: none">○ Platform○ Stopper																																																																													
10	Maße & Positionen	<table><tr><th>Element</th><th>Scale X</th><th>Scale Y</th><th>Scale Z</th><th>Pos X</th><th>Pos Y</th><th>Pos Z</th></tr><tr><td>Ground</td><td>23</td><td>1</td><td>23</td><td>0</td><td>0</td><td>0</td></tr><tr><td>Ground-Trigger</td><td>Ground* 0.95</td><td>Ground * 1</td><td>Ground* 0.95</td><td>0</td><td>Ground + 0.3</td><td>0</td></tr><tr><td>Walls (South and North)</td><td>25</td><td>30</td><td>1</td><td>0</td><td>15</td><td>+ -11</td></tr><tr><td>Walls (West and East)</td><td>1</td><td>30</td><td>25</td><td>+ -11</td><td>15</td><td>0</td></tr><tr><td>Avatar</td><td>1</td><td>2</td><td>1</td><td>0 (Start)</td><td>3 (Start)</td><td>0 (Start)</td></tr><tr><td>Distractor</td><td>1</td><td>1</td><td>1</td><td>Platform+0</td><td>Platform+1</td><td>Platform+0</td></tr><tr><td>Platform</td><td>5</td><td>1</td><td>5</td><td>7 (first Platform)</td><td>1 (first Platform)</td><td>7 (first Platform)</td></tr><tr><td>Stopper 1</td><td>5</td><td>3</td><td>1</td><td>Platform+0</td><td>Platform+1,5</td><td>Platform+ - 2</td></tr><tr><td>Stopper 2</td><td>1</td><td>3</td><td>5</td><td>Platform+ - 2</td><td>Platform+1.5</td><td>Platform+0</td></tr><tr><td>Platform-Trigger</td><td>Platform * 0.9</td><td>Platform * 1</td><td>Platform * 0.9</td><td>Platform+0</td><td>Platform+0,1</td><td>Platform+0</td></tr></table>	Element	Scale X	Scale Y	Scale Z	Pos X	Pos Y	Pos Z	Ground	23	1	23	0	0	0	Ground-Trigger	Ground* 0.95	Ground * 1	Ground* 0.95	0	Ground + 0.3	0	Walls (South and North)	25	30	1	0	15	+ -11	Walls (West and East)	1	30	25	+ -11	15	0	Avatar	1	2	1	0 (Start)	3 (Start)	0 (Start)	Distractor	1	1	1	Platform+0	Platform+1	Platform+0	Platform	5	1	5	7 (first Platform)	1 (first Platform)	7 (first Platform)	Stopper 1	5	3	1	Platform+0	Platform+1,5	Platform+ - 2	Stopper 2	1	3	5	Platform+ - 2	Platform+1.5	Platform+0	Platform-Trigger	Platform * 0.9	Platform * 1	Platform * 0.9	Platform+0	Platform+0,1	Platform+0
Element	Scale X	Scale Y	Scale Z	Pos X	Pos Y	Pos Z																																																																									
Ground	23	1	23	0	0	0																																																																									
Ground-Trigger	Ground* 0.95	Ground * 1	Ground* 0.95	0	Ground + 0.3	0																																																																									
Walls (South and North)	25	30	1	0	15	+ -11																																																																									
Walls (West and East)	1	30	25	+ -11	15	0																																																																									
Avatar	1	2	1	0 (Start)	3 (Start)	0 (Start)																																																																									
Distractor	1	1	1	Platform+0	Platform+1	Platform+0																																																																									
Platform	5	1	5	7 (first Platform)	1 (first Platform)	7 (first Platform)																																																																									
Stopper 1	5	3	1	Platform+0	Platform+1,5	Platform+ - 2																																																																									
Stopper 2	1	3	5	Platform+ - 2	Platform+1.5	Platform+0																																																																									
Platform-Trigger	Platform * 0.9	Platform * 1	Platform * 0.9	Platform+0	Platform+0,1	Platform+0																																																																									

		 <p>(Top-Down Ansicht Skizze mit möglichen X und Y Positionierungen der Plattformen. Y-Koordinate variabel)</p>
11	Event-System	<ul style="list-style-type: none"> - Aufgerufen durch einfache Events <ul style="list-style-type: none"> ○ <code>hdlPauseReturn()</code> → Main.ts Z.715 <ul style="list-style-type: none"> ▪ Wird ausgelöst durch einen Button der Gui und lässt von dem Pause-Screen zurück zum Spiel gehen ○ <code>hdlStart()</code> → Main.ts Z.752 <ul style="list-style-type: none"> ▪ Wird ausgelöst durch einen Button der Optionen und lässt das Spiel laufen - Aufgerufen durch <code>fc.EventPhysics</code> <ul style="list-style-type: none"> ○ <code>hdlGroundTrigger(_event: fc.EventPhysics)</code> → Main.ts Z.341 - Aufgerufen durch <code>KeyboardEvent</code> <ul style="list-style-type: none"> ○ <code>function hdlJump(_event: KeyboardEvent)</code> → Main.ts Z.404 ○ <code>hdlJump(_event: KeyboardEvent)</code> → Avatar.ts Z.88 - Aufgerufen durch <code>fc.Eventpointer</code> <ul style="list-style-type: none"> ○ <code>hdlCollision = (_event: fc.EventPointer)</code> → <code>bounceComponentScript.ts</code> Z.15 - Durch mehrere "<code>fc.Keyboard.isPressedOne()</code>"-Funktionen wird die komplette Steuerung des Avatars geregelt (Main.ts, <code>listenForKeys()</code>, Z. 242)
12	Eigene Component	<ul style="list-style-type: none"> - Die Klasse „ComponentBounce“ ist eine Erweiterung der ComponentScript-Klasse von Fudge. Sie wird jedem Distractor zugewiesen. Sie bewirkt, wenn der Avatar den RigidBody des Distractors berührt, dass ein Sound abgespielt, der Avatar paralysiert und weggestoßen wird