UNIKASSEL VERSITÄT **Universität Kassel**
FB Elektrotechnik / Informatik
FG Technische Informatik
Prof. Dr. Ing. D. Wloka
Dipl. Inf. Christian Niemand
WS 2016/17

FACHGEBIET TECHNISCHE INFORMATIK

Pflichtenheft
„Avatare in Serious Games“

Abschlussprojekt

Hausarbeit dient als Leistungsnachweis zur Veranstaltung

Projektbezeichnung: Abschlussprojekt Avatare in Serious Games
Matrikelnummer.: 30204300
Name: Christian Wiegand
Datum: 31.03.2017

1 Allgemeine Arbeits- und Bewertungsrichtlinien

Im Folgenden werden allgemeine Bewertungsrichtlinien für das Projekt festgelegt. Jedes Kriterium kann in der speziellen Projektbeschreibung angepasst und konkretisiert werden. Es kann vorkommen, dass einzelne Richtlinien nicht in Ihrem speziellen Projekt angewendet werden können. Die Anzahl der zu vergebenen Punkte entnehmen Sie bitte der Projektbeschreibung und erarbeiten die Details mit Ihrem Betreuer.

Bitte beachten Sie, dass die angegebenen Bewertungsrichtlinien und Vorgaben erfüllt werden müssen, um die Veranstaltung als bestanden zu absolvieren. Für gute bis sehr gute Ergebnisse wird ein Eigenanteil erwartet, der nicht als Vorgabe ausformuliert ist und von Ihnen individuell festgelegt werden kann.

1.1 Vorarbeiten

Erstellen Sie geeignete Assets (Texturen, Modelle, Sounds, etc.), die Ihnen als Platzhalter dienen können und dokumentieren Sie deren Formate sowie Metadaten.

1.2 Programm-Versionen:

Es sind folgende Versionen zu verwenden. Nur diese Software-Versionen sind auf dem Testrechner installiert. Alle Modelle, Assets, usw. die damit nicht geöffnet werden können, werden als nicht vorhanden gewertet.

- Autodesk Suite 2016, d.h. 3ds Max 2016, Mudbox 2016, ...
- Unity 5.5.x
- Mono (Unity) oder Visual Studio 2015

1.3 Projektverzeichnisse

Wählen Sie eine sinnvolle Verzeichnisstruktur. Erstellen Sie ggf. Unterordner z.B. für Skripte, Texturen, Sounds, Prefabs etc.

1.4 Einheiten

In allen Tools zur Modellierung soll (sofern möglich) Meter als Standard Einheit verwendet werden. Somit ist die Verwendung von Objekten in allen Tools ohne Konvertierung möglich.

1.5 Relative Pfade

Verwenden Sie immer relative Pfade. Dies ist wichtig, weil Ihre Projekte ansonsten auf anderen Rechnern nicht korrekt ausgeführt werden können, da bestimmte Ressourcen nicht an entsprechender (absoluter) Stelle gefunden werden können.

1.6 Explorer Vorschau

Erzeugen Sie von jedem selbst modellierten Objekt bzw. jeder Szene eine gerenderte Vorschau oder fertigen Sie ein aussagekräftigen Screenshot aus dem Tool heraus an, sodass man bereits im Explorer eine schnelle Vorschau zu den Projektinhalten bekommen kann.

1.7 Modularisierung vs. Gesamtprojekt

Für jedes Tool muss ein Gesamtprojekt vorliegen. Um jedoch eine möglichst hohe Wiederverwendbarkeit zu gewährleisten, sollten komplexere Szenen bzw. Objekte sinnvoll modularisiert werden. Wenn Sie z.B. eine Stadt bauen, bietet es sich an einzelne Gebäude in jeweils einer separaten .max Datei zu speichern. Für Inneneinrichtung gilt das Gleiche. Auf diese Weise können z.B. Möbel in unterschiedlichen Häusern und Häuser in verschiedenen Städten verwendet

werden.

1.8 Strukturierung

Achten Sie bei der Umsetzung auf sinnvolle Strukturen und Namenskonventionen.

1.9 Formales

Während des Bearbeitungszeitraums wird ein Statustermin angeboten, indem der aktuelle Stand sowie Probleme besprochen werden können. Zugang zum Studenten-Rechner kann jederzeit von den Mitarbeitern des Fachgebiets erfragt werden.

Nach der Abgabe erfolgt eine Präsentation der Ergebnisse, zur Überprüfung, ob die Arbeit selbstständig erstellt wurde. Zur Teilnahme an dem Projekt als Prüfung ist die rechtzeitige Anmeldung bei der „Online Klausuranmeldung“ des FB16 notwendig. Der dort angegebene Termin ist gleichzeitig auch der finale Abgabetermin.

1.10 Dokumentation

Dokumentieren Sie Ihre Arbeit mit Hilfe dieses Dokuments. Tragen Sie auf der Titelseite Ihren Namen, Matrikelnummer und die Projektbeschreibung ein. Fügen Sie Ihre Dokumentation hinter den Pflichtenheft Teil ein. Achten Sie darauf, dass die Haupt-Nummerierung der Aufgabenstellung erhalten bleibt! Das ausgedruckte Dokument und alle erstellten Daten geben Sie zum Schluss auf CD/DVD ab, diese gehören zur Prüfungsleistung. Dieses Dokument stellt den Leitfaden der Arbeit dar.

1.10.1 Doxygen

Um die Autorschaft sowie die Schnittstellen Ihres Quelltextes zu dokumentieren verwenden Sie bitte Doxygen Kommentare. Dokumentieren Sie das Datum der Ersterstellung.

1.11 Coding Guidelines

- Bitte keine Tabs sondern Leerzeichen verwenden. Das kann im Editor eingestellt werden.
- Bitte CamelCase verwenden und keine _ zur Worttrennung.

1.12 Abschließende Hinweise

- Lesen Sie zunächst das gesamte Pflichtenheft durch!
- Verwenden Sie gegebenenfalls Ihnen zur Verfügung gestellte Materialien sowie am Fachgebiet vorhandene Soft- und Hardware.
- Recherchieren Sie zu möglicherweise bereits vorhandenen, nutzbaren Konzepten oder Techniken, die Ihnen bei der Bearbeitung helfen können. Geben Sie dabei unbedingt die Quellen vollständig an. „Aus dem Internet“ ist keine Quellenangabe und muss als Plagiatsversuch gewertet werden!
- Beginnen Sie die Bearbeitung, indem Sie die einzelnen Aufgabenstellung überprüfen, sich eine Herangehensweise zurechtlegen und Ihren angedachten Workflow mit einfachen Modellen überprüfen und dokumentieren.
- Die finale Präsentation findet auf dem Studenten-Rechner des Fachgebietes statt. Testen Sie den Datenträger und ihr Projekt am besten dort (siehe nächster Punkt).
- Die Bewertung erfolgt ausschließlich auf den Daten/Programmen/Modellen des eingereichten Datenträgers, die auf dem Fachgebietsrechner funktionieren.

2 KI Für Avatare und Spielelemente und Bedienung

Beschreibung der Vorgehensweise und der Techniken

Idle Verhalten:

1. Navigieren auf einem Navigation Mesh und Hindernissen ausweichen (10p)

   Das Navigation Mesh bietet Unity an und kann auf der Geometrie generiert werden.

   Dieser Punkt ist zwingend erforderlich, weil sich die Avatare sonst nur schwer korrekt und realistisch bewegen lassen.

   - NavMesh baken (10p)

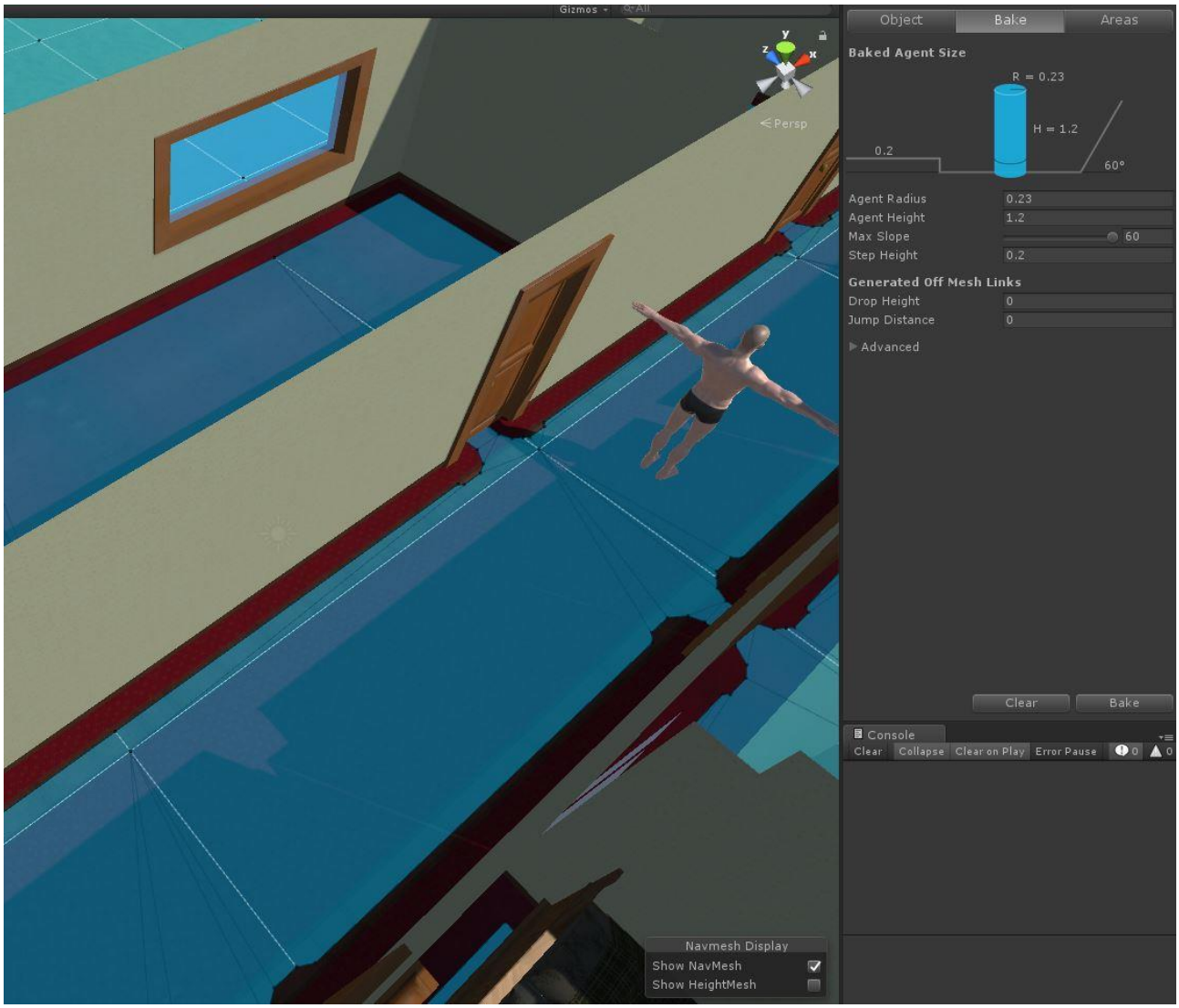


Abbildung 1: NavMesh und seine Einstellungen

Lösung:

Um das NavMesh zu baken genügt ein Klick auf „Bake" (siehe Abb. 1). Damit bei der Treppe auch ein begehbares NavMesh generiert wird, habe ich den Agent Radius von 0.23 gewählt. Ich habe festgestellt, dass im rechten Treppenhaus eine Lücke im NavMesh war und dass daher kein durchgängiges Mesh

generiert wurde. Das habe ich gelöst indem ich einen unsichtbaren Cube dazwischen platziert habe. Anschließend wurde das Mesh durchgängig darüber generiert. Im Laufe der Entwicklung hat sich herausgestellt, dass unter manchen Objekten ein NavMesh sein sollte, damit mit diesen gut interagiert werden kann. Diese sind z.B. die Sofas, die Betten und die Türen. Über das Einstellen von „Nothing" bei „Static" ist das auch gelöst (siehe Abb. 2).

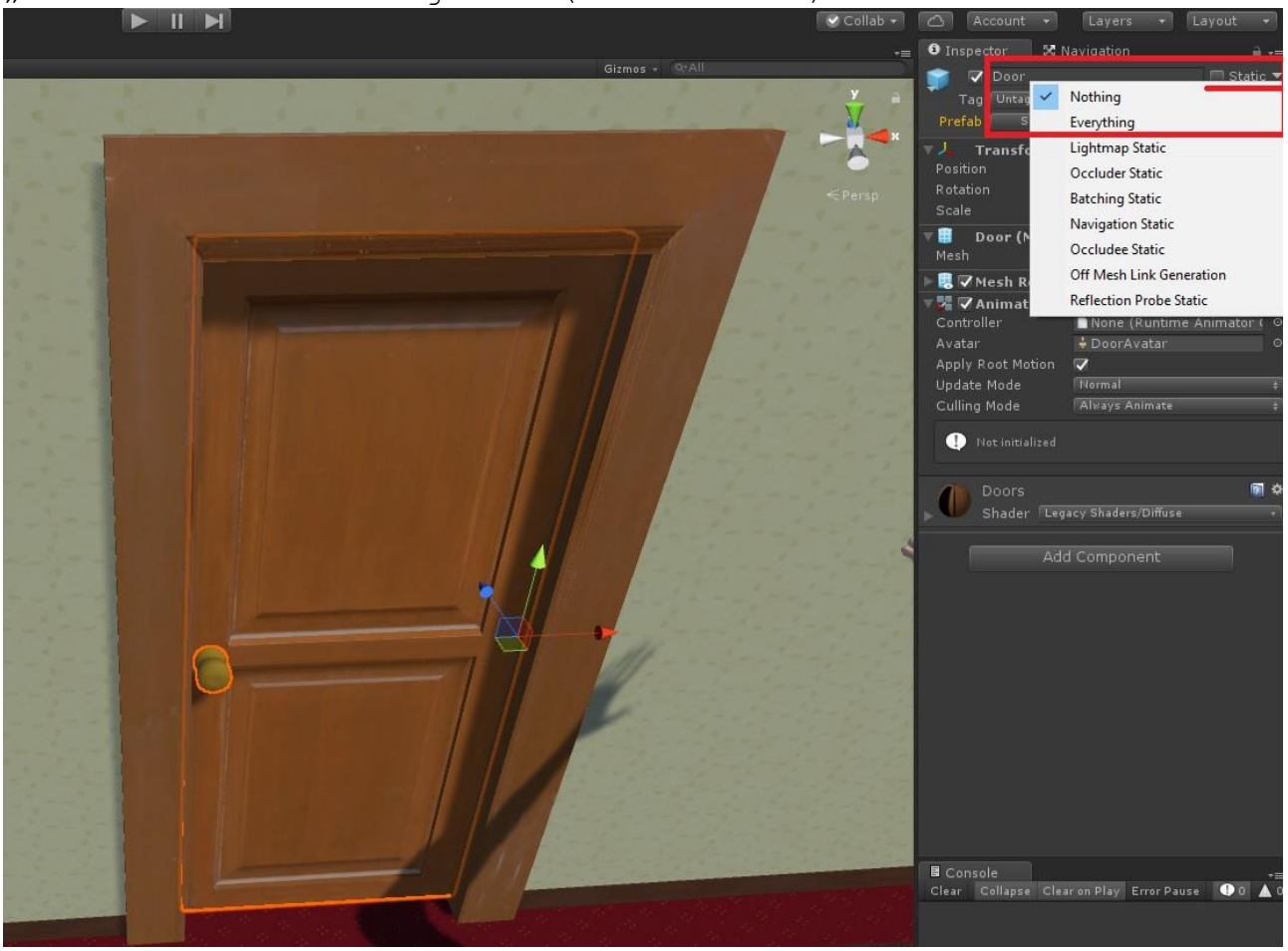


Abbildung 2: nicht Static

2. Avatar soll zufälligen Tasks nachgehen (z.B. auf Sofa sitzen, Bild anschauen, aus Fenster sehen, auf Zimmer gehen) (15p)

   Für diesen Punkt sind ein Animator und ein Skript notwendig. Der Animator mit allen Animationen, die aktiviert werden können. Das Skript soll dann zufällig die nächste Tätigkeit bestimmen und ggf. den Avatar dorthin gehen lassen (z.B. Sofa). Zwingend erforderlich ist dieser Punkt nicht, denn die Avatare könnten auch einfach herumstehen bis ein Feuer ausbricht. Abhängig ist dieser Punkt vom Navigation Mesh.

   - Auf Sofa sitzen (5p)

Abbildung 3: Avatar sitzend auf dem Sofa

Lösung:

Über das an den Avatar angehängte „Activity Controller" Skript kann man im Inspector eine Aktivität einstellen. Für die aktuell ausgewählte Aktivität werden zunächst alle sich auf der Etage befindenden Ziele geladen und falls es welche gibt wird dieses als Ziel für setDestination() gesetzt. Ich habe es so gelöst, dass auf den Etagen unsichtbare Cubes als Ziele verteilt sind. Damit sie als Ziel korrekt erkannt werden, müssen sie getaggt sein mit der Aktivitätsbezeichnung + der Etagenzahl. Durch das setzen des Animatortriggers für „gehen" geht er in die gehende Animation über. Dort angekommen wird

der NavMeshAgent gestoppt und der Avatar per Quaternion.slerp() so gedreht, dass er sich hinsetzen kann, also mit dem Rücken zum Sofa in diesem Fall (siehe Abb. 3). Wenn das soweit ist, wird die „setzen" Animation angestoßen. Damit ist der Vorgang nun abgeschlossen. Bei der Lösung für das NavMesh habe ich erwähnt, dass u.a. das Sofa nicht Navigation Static sein sollte. Ich habe es zu Beginn versucht mit statischen Objekten, aber dann wollte der Avatar sich nie richtig hinsetzen (immer zu weit weg). Daher habe ich es so gelöst, dass es durchgängig ist. Aber dadurch dass die Avatare nur von Destination zu Destination gehen, gehen sie nur im Ausnahmefall durch solche Objekte hindurch.

- Bild anschauen (5p)

Abbildung 4: Avatar betrachtet ein Gemälde

Lösung:

Zuvor saß der Avatar auf dem Sofa, wenn man seine Activity nun über den Inspector auf „Bild“ ändert, dann muss er zunächst wieder aufstehen. Dies geschieht dadurch, dass im Animator das Bool für die Activität deaktiviert wird und für die neue Aktivität aktiviert wird.

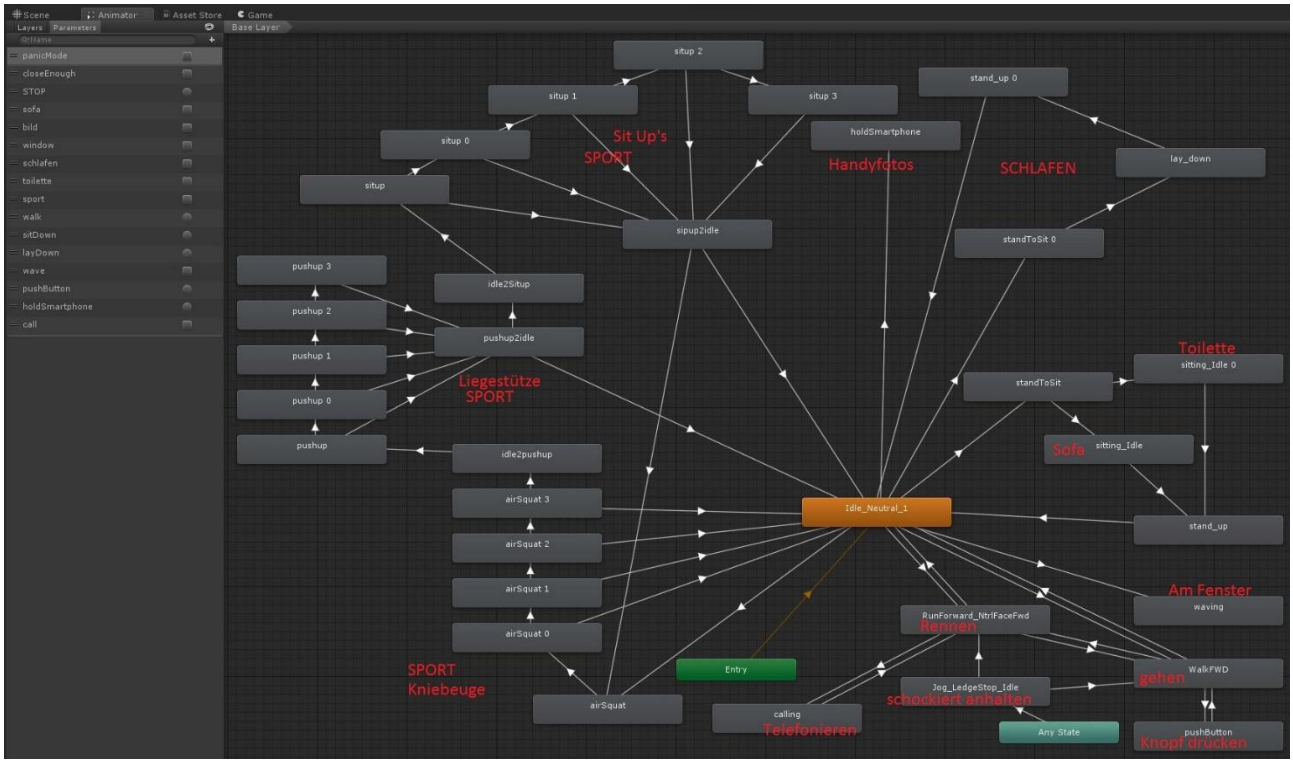


Abbildung 5: Animator

Dadurch geht er zu „stand_up“ und von dort zu „idle“ und durch die Aktivierung der neuen Aktivität zu „WalkFWD“ (siehe Abb. 5). Dann begibt er sich zu einem Gemälde-Ziel und bleibt davor stehen. Somit sieht es so aus, als würde er es angucken (siehe Abb. 4).

- Aus dem Fenster sehen (5p)

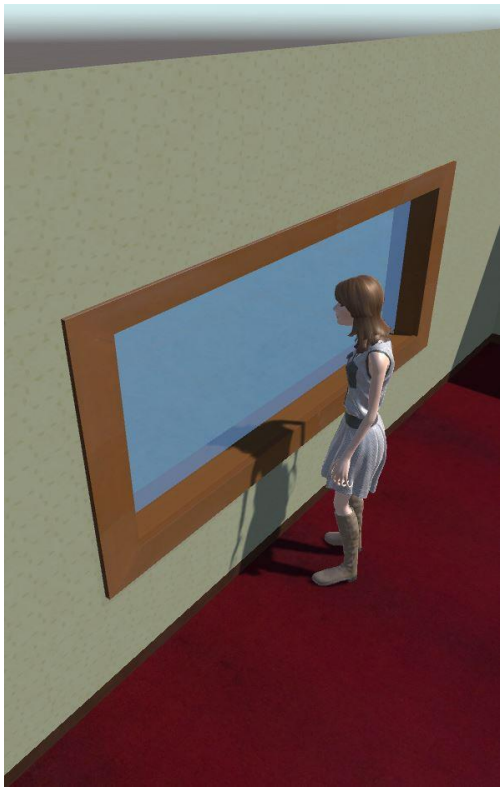


Abbildung 6: Avatar sieht aus dem Fenster

Lösung:
Diese Aktivität ist genauso aufgebaut wie die „Gemälde"-Aktivität. Der Avatar geht zum Ziel und bleibt davor stehen. (siehe Abb. 6).

3. Auf dem Zimmer schlafen (5p)

Theoretisch reicht es für diesen Punkt aus, wenn ein Avatar eine schlafende Animation hat. Realistischerweise sollten Avatare aber auch aufwachen falls es Feueralarm gibt (bei Rauch nicht unbedingt). In diesen Fall muss im Controllerscript angestoßen werden, dass er in den Wachzustand übergeht mit anschließendem fliehen. Wenn es ein Avatar mit sehr tiefem Schlaf ist, ist diese Tätigkeit von keinem der Punkte Abhängig. Sonst von Punkt 1. Zwingend erforderlich ist es nicht, dass Avatare schlafen.

Abbildung 7: Schlafender Avatar

Lösung:

Bei dieser Aktivität geht der Avatar zunächst vor das Bett, setzt sich drauf und legt sich dann hin. Leider liegt er dann nur zu ¾ auf dem Bett, also habe ich ihm dabei noch einen Schubs gegeben per Vector3.Lerp(), was allerdings auch nicht immer funktioniert. Daher setze ich nach einem Moment die Position fest auf das Ziel (siehe Abb. 7). Beim Aufstehen gibt es auch noch Verbesserungspotential, denn es kommt vor, dass der Avatar „im" Bett aufsteht, statt an der Kante.

4. Individuelles Verhalten (10p)

Toilettengang (5p): Dieser Zustand ist dem Schlafen ähnlich, denn der Avatar kann nicht sofort weg, kriegt dafür aber alles gleich mit. Per Skript muss nach dem feststellen des Feueralarms oder eines Feuers eine Animation abgespielt werden die den Abschluss des Toilettengangs durchführt. Danach soll der Avatar fliehen. Dieser Punkt ist nicht zwingend erforderlich und hängt von nichts ab.

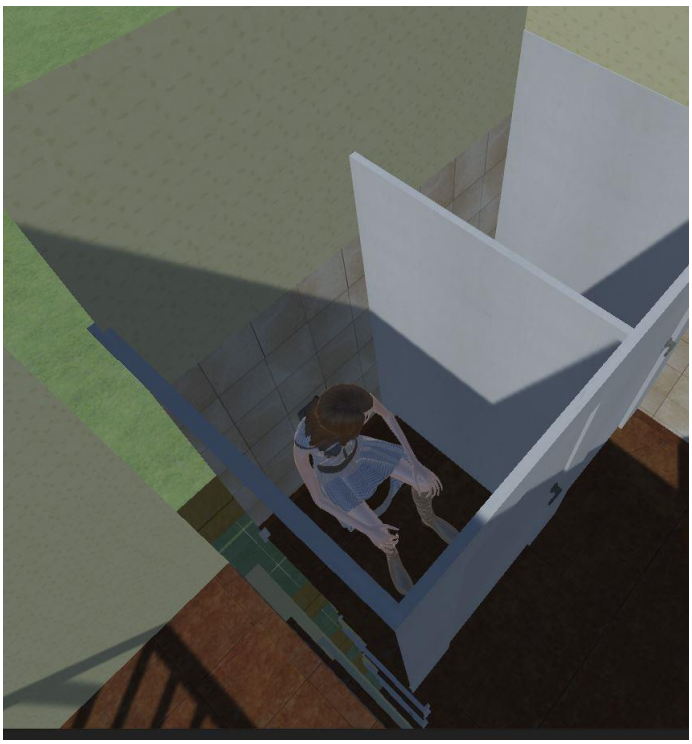
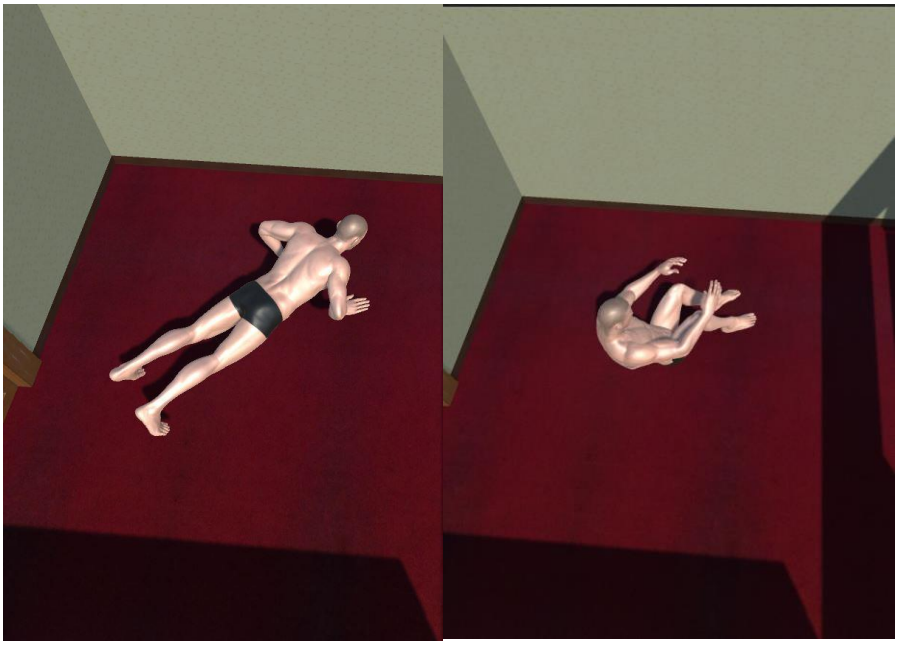


Abbildung 8: Avatar auf der Toilette

Lösung:

Dies habe ich ähnlich gelöst wie das setzen auf das Sofa. Der Avatar geht zum Ziel, dreht sich mit dem Rücken zum Ziel, und setzt sich hin (siehe Abb. 8). Beim Wechsel des Zustands steht er wieder auf.

Sport (5p): Hier macht der Avatar gerade sportliche Übungen (z.B. Liegestütze, Kniebeugen). Dafür geht er nach einander alle Animationen durch bis er ein Feuer oder den Feueralarm feststellt. Danach flieht er. Dieser Punkt ist nicht zwingend erforderlich und hängt höchstens vom Navigation Mesh ab, damit der Avatar nicht an unsinnigen Orten Sport treibt.



Abbildungen 9 und 10: Liegestütze und Situp's

Lösung:

Hierbei sucht der Avatar erstmal wieder einen geeigneten Ort auf. Anschließend beginnt er mit seinen Übungen. Zuerst Kniebeuge, dann Liegestütze und danach Situp's (siehe Abb. 9 und 10). Falls nichts dazwischen kommt, beginnt er dann wieder von vorn. Falls doch etwas dazwischen kommt geht er in eine Übergangsanimation und geht danach in „idle" über (siehe Abb. 5).

Panik Verhalten:

1. Fluchtweg suchen – Fahrstuhl vermeiden – Sammelbereich aufsuchen (10p)

In einem Skript über die Objekt Koordinaten ausrechnen lassen welches Treppenhaus gerade näher ist. Dann kann sich der Avatar per setDestination() auf dem NavMesh dorthin und anschließend zum Sammelbereich bewegen. Der Fahrstuhl wird ohnehin vermieden, da ich ihn nicht implementiere. Dieser Punkt ist zwingend erforderlich und vom NavMesh abhängig.

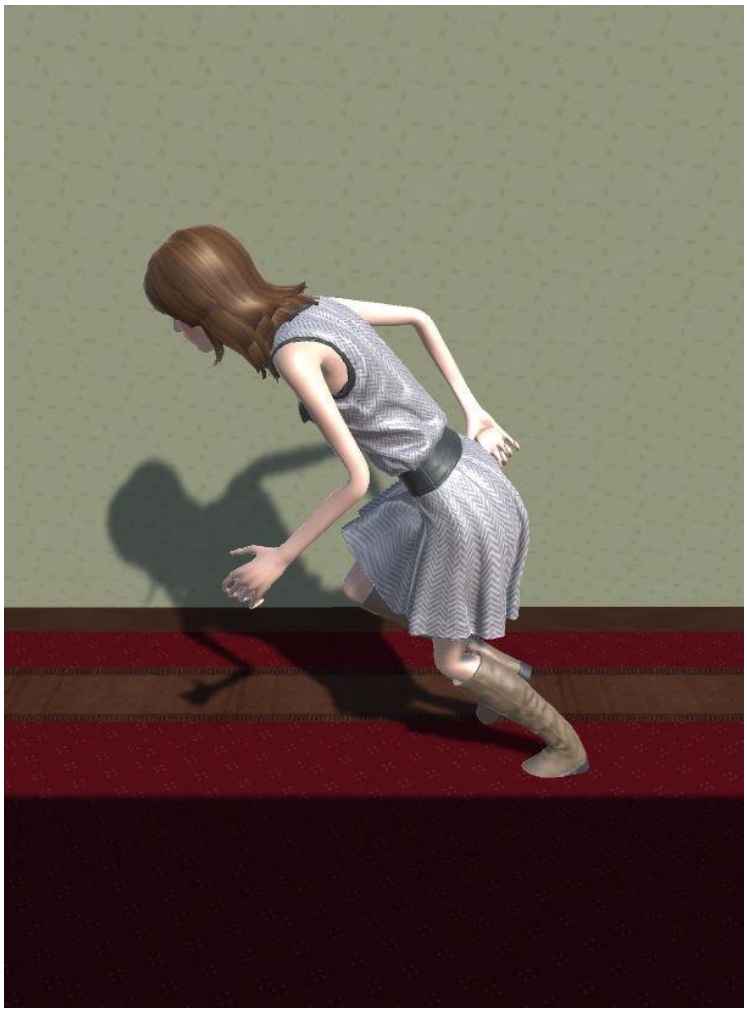


Abbildung 11: Avatar flieht

Lösung:

Um einen Fluchtweg zu finden genügt es, dem Avatar ein Destination zu geben. Unter normalen Umständen (also wenn

kein Feuer in die Quere kommt) sucht sich der Avatar zuerst das am nächsten liegende Treppenhaus auf (siehe Abb. 11). Er findet dieses indem er alle Destinations dieser Etage sucht (per Tag) und sie mit einer SortedList sortiert. Der erste Eintrag ist das am nächsten gelegene Ziel. Nach Erreichen dieses Ziels geht er zum Sammelpunkt weiter (Destination vor dem Gebäude). Die Fluchtziele müssen auch alle korrekt getaggt sein mit „destFloor" + Etagenzahl.

2. Feuer erkennen und vermeiden, Alarm wahrnehmen (20p)

   Um Feuer zu erkennen kann man zwei Methoden miteinander kombinieren. Zum einen mit einem Sichtkegel vom Avatar aus, und zum anderen durch einen Umkreis um das Feuer. Schließlich würde man ein Feuer auch durch die Wärme spüren. Um das Feuer zu vermeiden könnte von dem Avatar aus nach vorne ein Raycast gemacht werden, der prüft, ob vor ihm ein Feuer ist. Wenn dies der Fall ist und das Feuer nah genug ist, dann soll er eine alternative Destination wählen. Für die Wahrnehmung des Alarms bietet sich an auf einer Etage einfach eine gewisse Entfernung von einer Alarmquelle als „Hörradius" auszuwählen und Avatare die sich außerhalb befinden, hören ihn dementsprechend nicht. Dieser Punkt ist zwingend erforderlich. Dieser Punkt erfordert, dass es Feuerquellen und Feueralarme gibt.

   - Avatare erkennen per Sichtkegel Feuer (5p)

Abbildung 12: Avatar erkennt ein Feuer per Sichtkegel

Lösung:

Per Raycast wird ein gewisser Bereich vor dem Avatar überprüft (einstellbar am Skript)(siehe Abb. 12). Wenn ein Feuer entdeckt wird, wird die Methode burn() im AvatarController aufgerufen. Das Skript für die Aktivitäten deaktiviert sich anschließend. Damit Feuer erkannt werden müssen diese den Layer „Fires" haben. Und am FieldOfView Skript muss dieses als Target Mask eingestellt sein. Außerdem sollte das Gebäude das Layer „Walls" bekommen, damit es ebenfalls am FieldOfView Skript eingestellt werden kann als Obstacle Mask. Sonst erkennt der Avatar Feuer durch die Wand.

- Avatare erkennen Feuer in unmittelbarer Nähe (5p)

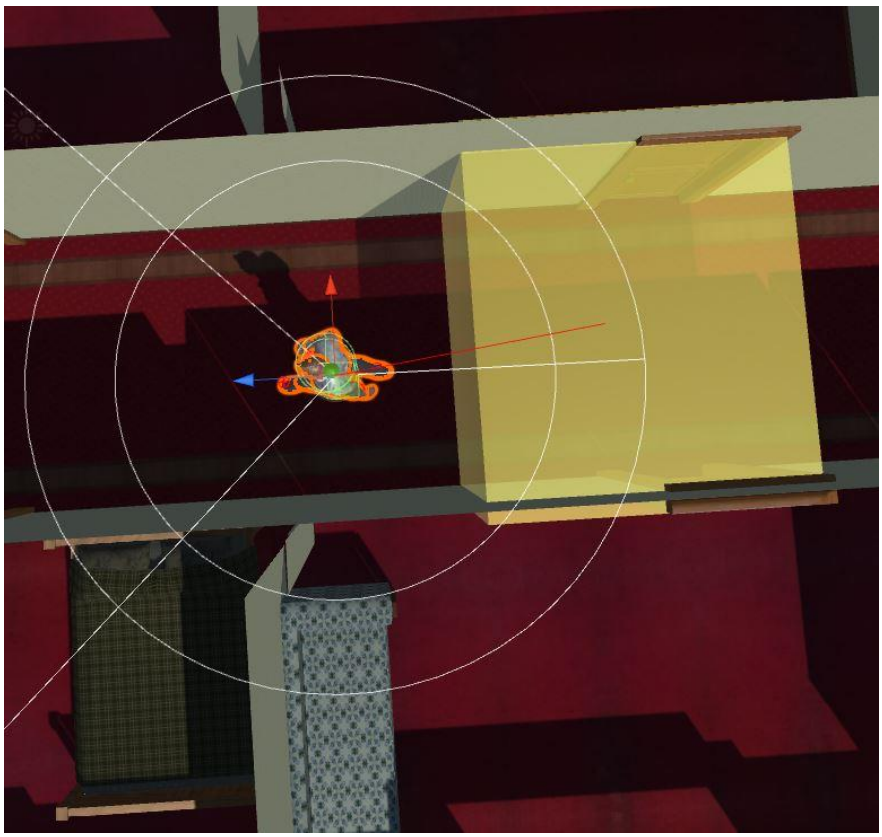


Abbildung 13: Avatar bemerkt ein Feuer im Nahbereich hinter sich

Lösung:

Der innere Kreis der auf Abb. 13 zu sehen ist, ist für den Nahbereich um den Avatar zuständig. Er erkennt Feuer 360° um ihn herum. Er simuliert das Fühlen. Auch hier wird bei Erkennung „burn()" aufgerufen und jede Aktivität eingestellt.

- Avatare wählen bei Feuer andere Destination (5p)

Abbildung 14: Der weibliche Avatar wählte das andere Treppenhaus

Lösung:

Als ein Beispiel führe ich hier die Situation in Abb. 14 an. Zum Zeitpunkt an dem der weibliche Avatar das Feuer im Treppenhaus erkannte wählte sie wie oben beschrieben zunächst das Ziel des rechten Treppenhauses aus. Da dieses aber im Feuer oder dahinter liegt wählte sie das andere Treppenhaus aus, auch wenn der Weg länger ist. Ich unterscheide hier normale Feuer, Raumfeuer und Treppenhausfeuer (über den Namen). Treppenhausfeuer bewirken, dass der Avatar das Treppenhaus wechselt. Die Ziele im Treppenhaus sind so gewählt, dass der Avatar nicht durch das Feuer läuft, sondern beim Erkennen zurückläuft und das Treppenhaus wechselt. Falls der Avatar bereits im Treppenhaus ist und auf einer anderen als seiner ursprünglichen Etage ein Feuer entdeckt, lädt er die Fluchtziele der momentanen Etage und wählt das Treppenhausziel im anderen Treppenhaus.
Beim Erkennen eines Raumfeuers aktiviere ich für einen kurzen Moment ein navMeshObstacle am Feuer, damit der Avatar am Feuer vorbei flieht und nicht durchläuft.
Beim Erkennen eines sonstigen Feuers, wählt der Avatar einfach das nächstgelegene Ziel in seiner Zieleliste, falls dieses nicht vom ihm aus hinter dem Feuer liegt.

- Avatare nehmen den akustischen Alarm wahr (5p)

Abbildung 15: Durch den Feueralarm fliehen alle Avatare

Lösung:

Sobald der akustische Alarm aktiv ist, gehen alle Avatare in den Panikmodus über (siehe Abb. 15). Der Alarm wird erkannt durch eine Überprüfung des Alarmtextfelds. Ich habe mich dagegen entschieden „Hörreichweiten" einzubauen, da man einen Feueralarm normalerweise im gesamten Gebäude deutlich hören können sollte. Panikmodus bedeutet, dass die Avatare ihr Aktivitätsskript beenden und das Fluchtskript ausführen mit Beginn bei Start().

3. Feuermelder betätigen (sollte ein Feuer nicht entdeckt werden, geht der Alarm nach 30 Sekunden automatisch an) (5p)

   Falls ein Avatar ein Feuer bemerkt und keinen Alarm wahrnimmt, schaut er ob ein Feuermelder in seiner näher ist, sonst flieht er. Da man annehmen kann, dass sich der Avatar in seiner näheren Umgebung auskennt, überprüft man per Skript, ob es innerhalb eines gewissen Umkreises einen Feuermelder gibt. Wenn ja, dann bewegt er sich dort hin und macht eine „Drück"-Animation in Richtung des Knopfs. Danach ist der Alarm an. Dieser Punkt ist teilweise erforderlich, eigentlich geht der Alarm ohnehin nach 30 Sekunden an. Dieser Punkt ist wieder vom NavMesh abhängig und davon, dass Avatare Feuer und Alarme wahrnehmen können.

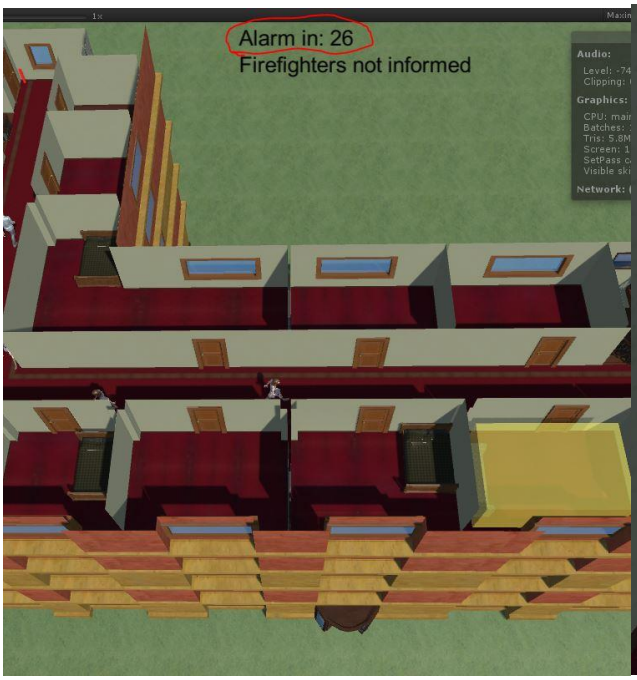


Abbildung 16: Wenn ein Feuer ausbricht, läuft der Alarmtimer. Abb. 17: Avatar betätigt den Feueralarm

Lösung:

Durch betätigen des Buttons für das Hinzufügen eines zufälligen Feuers, wird als Seiteneffekt der Alarmtimer gestartet (siehe Abb. 16).

Falls ein Avatar ein Feuer entdeckt, während es keinen Alarm gibt, wird er zunächst fliehen, aber auf der Flucht Ausschau nach Feuermeldern halten. Wenn er einen erkennt (das dritte FielOfView-Skript, mittlerer Kreis) wird er dort einen Zwischenstopp einlegen und ihn betätigen (siehe startAlarm()). Dazu wird kurzzeitig ein neues Destination gesetzt, aber das Alte gemerkt. Angekommen wechselt der Avatar in die „drück" Animation (siehe Abb. 17), der Alarm geht los und der Avatar setzt seinen alten Weg fort.

4. Aus dem Fenster winken, falls Fluchtweg versperrt ist (5p)

Zuerst muss im AvatarController eine Liste mit möglichen FluchtDestinations gespeichert werden. Im Feuerfall werden diese der Entfernung nach abgearbeitet, d.h. der Avatar würde den nächstgelegenen Fluchtweg bevorzugen. Falls aber keine mehr übrig sind, würde er wenn möglich als nächste Destination ein Fenster wählen und dort in eine „Wink"-Animation übergehen.

Dieser Punkt ist nicht zwingend erforderlich und Abhängig vom NavMesh.

Abbildung 18: Avatar steht am Fenster und winkt

Lösung:
Wie bereits weiter oben erwähnt hat der Avatar beim Fliehen auf einer Etage (also bevor er im Treppenhaus ankommt) eine Liste von Destinations. Wenn er auf Feuer trifft arbeitet er diese ab. Wenn es nun der Fall ist, dass er alle abgearbeitet hat, dann lädt er alle Fensterziele auf dieser Etage und wählt das nächste Fenster aus. Dann setzt er dieses als Ziel und bewegt sich dort hin. Falls er auf dem Weg wieder auf ein Feuer trifft, wechselt er zum nächsten Fensterziel. Am Fenster angekommen stoppt er wieder den NavMeshAgent und setzt beim Animator das Bool für „winken" (siehe Abb. 18).

5. Gaffen bzw. Fotos mit dem Handy machen (5p)

   Wahrscheinlich werden Avatare erstmal fliehen bevor sie gaffen. Daher bietet es sich an, wenn der Avatar am Sammelpunkt ist und noch ein Feuer sehen kann, er anfängt zu gaffen und Handyfotos zu machen. Hier kann wieder die eben genannte Feuererkennung benutzt werden (evtl. etwas erweitert). Dieser Punkt ist nicht zwingend erforderlich und abhängig von der Feuererkennung.

Abbildung 19: Manche Avatare machen Handyfotos, manche „gaffen“ nur herum

Lösung:

Als „gaffen“ habe ich es gelten lassen, dass die Avatare draußen stehen und herumgucken. Per Random.Range() lasse ich entscheiden, ob sie gaffen oder Handyfotos machen (siehe Abb. 19). Für die Animation „Handyfoto machen“ habe ich einen Smartphoneförmigen Cube unter den Parentknoten des Avatars gehängt. Dieser ist nur für das Handyfoto machen zuständig. Für das Notruf-rufen gibt es ein anderes Handy, damit es leichter ist diese anzuzeigen. Erst beim Start der „Handyfoto machen“-Animation wird das Smartphone angezeigt.

6. Notruf anrufen (15p)

Wenn ein Avatar gerade flieht und sich nicht in Lebensgefahr befindet soll es sein können, dass er auf dem Weg den Notruf wählt. Dafür benutzen die Avatare ihr Handy. Nach dem ersten wählen erscheint die Feuerwehr vor dem Gebäude. Bonus: Wenn ein Avatar den Notruf wählen will und andere um ihn herumstehen, dann soll er die Anderen fragen, ob sie den Notruf bereits gewählt haben. Dieser Punkt ist nicht zwingend erforderlich und von nichts abhängig.

- Avatare wählen auf der Flucht den Notruf (5p)

Abbildung 20: Avatar verständigt den Notruf

Lösung:

Im Zufallsverfahren entscheiden sich Avatare den Notruf zu wählen, wenn sie im Panikmodus sind (siehe Abb. 20). Nach dem ersten Notruf wird der Timer für die Feuerwehr am FireManager Skript gesetzt.

- Feuerwehr erscheint nach 2 min. (5p)

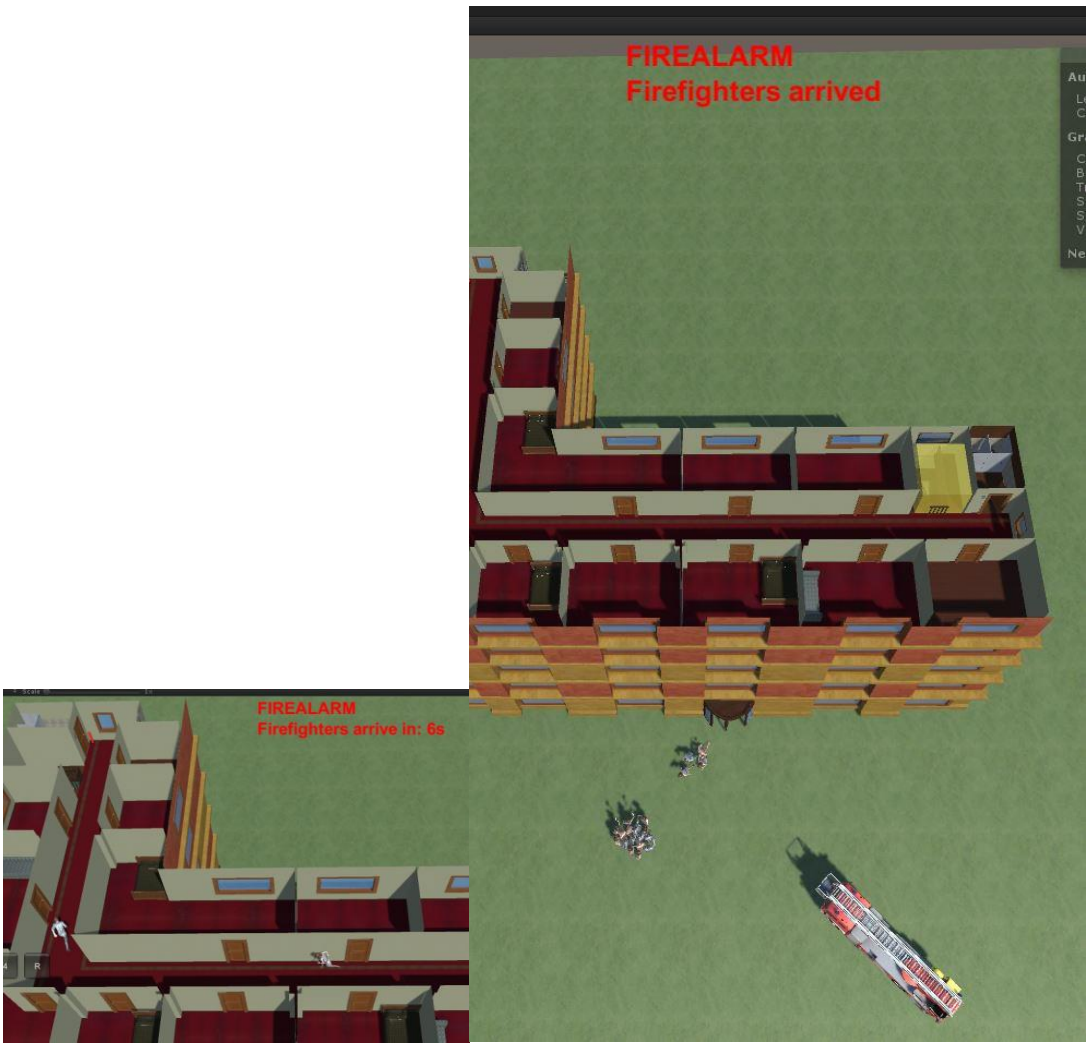


Abbildung 21: Feuerwehr wurde informiert, Abb. 22: Die Feuerwehr ist angekommen

Lösung:

Wenn der Timer für die Feuerwehr am FireManager Skript gesetzt wurde (siehe Abb. 21) und abgelaufen ist, lasse ich den eigentlich bereits vorhandenen Drehleiterwagen erscheinen (siehe Abb. 22). Alle Timer in dieser Simulation lasse ich per Coroutine laufen.

- Avatare sprechen sich ab (5p)

Lösung:

Diesen Punkt simuliere ich dadurch, dass die Avatare per Zufall genau so selten anrufen, dass es so aussieht als hätten sie sich abgesprochen.

Dokumentieren Sie Ihre Arbeit im Anhang Ihres Pflichtenheftes. Tragen Sie auf der Titelseite Ihren Namen, Matrikelnummer und die Projektbeschreibung ein. Fügen Sie während der Bearbeitung an den angegebenen Stellen Screenshots oder Bilder ein. Das ausgedruckte Dokument und alle erstellten Daten auf CD/DVD geben Sie zum Schluss ab, diese gehören zur Prüfungsleistung. Dieses Dokument stellt den Leitfaden der Arbeit dar. Zusätzlich zählt noch der Wiedererkennungswert der fertigen Teile.

Zum Bestehen sind **50%** der Punkte erforderlich. Insgesamt werden **100 Punkte (75 Pflicht + 25**

Bonus) verteilt. Alle Pflichtpunkte reichen für ein befriedigendes bis gutes Ergebnis (2,5). Durch hohe Qualität der Endergebnisse und durch individuelle Leistungen können Sie Bonuspunkte erlangen und ein sehr gutes Ergebnis (1,0) erreichen.

Checkliste: Abzugebende Medien

- ☐ Unity Projekt
- ☐ Alle Max-Dateien
- ☐ Alle Texturen
- ☐ Alle FBX-Dateien
- ☐ Alle Audio- und Beschreibungsdateien
- ☐ CD/DVD mit allen Dateien
- ☐ Ausgedruckte Ausarbeitung inkl. Screenshots