Aufgabe 1

Optimaler Ablauf:

- Man erhalte eine UIMA (Shapenet) Datei in XML Format **Example.xml**
- Man konvertiere diese in eine JSON Datei **Example.json**, die dann mit DeepSynth kompatibel ist
- Dies wird gespeichert in **%deep-synth%/shapenet/Example.json**
- Man führe deep-synth aus, mit dem Kommando
    - python batch_synth_shapenet.py --save-dir newtest_bar --data-dir living --model-dir res_1_living --location-epoch 300 --rotation-epoch 300 --start 0 --end 1 --shapenet-dir shapenet
- In **batch_synth_shapenet.py** wird **Example.json** eingelesen und dessen Raumgröße ermittelt. Ein ähnlich großer Raum wird dass aus SUNCG ausgesucht **Raum.json**.
- Diese Informationen werden an **scene_synth_shapenet.py** weitergegeben, was im Vergleich zur standarden **synth_shapenet.py**, eine leere Szene mit **Raum.json** als vorlage direkt mit Nodes aus **Example.json** belegt. Danach werden es, wie gewöhnlich, weitere SUNCG Objekte eingefügt, solange es der continue_predictor erlaubt.
- Eines der Ergebnisse wird mit dem Stichwort Final versehen z.B. **0_0_8_final.json**
- Das Importer-tool (Schritt12.java) sucht nach einer Datei mit so einem Stichwort und liest **0_0_8_final.json** dann ein, als Interne Repräsentation der Szene. Es ersetzt alle Objekte mit ihren ShapeNet Pendants.
- Nach dessen Durchlauf erhalte man eine Shapenet XML Datei **0_0_8_final.xml**, die standardmäßig mit der internen Speicherfunktion produziert wird

- **Verbesserungsidee**: Man behalte alle Objekte aus Example.xml und fügt diese in **0_0_8_final.xml** ein, damit zB. eine rote Couch nicht durch eine blaue ersetzt wird, kraft der Konvertierung con XML – JSON- und wieder XML. Es ist besser Originale zu behalten als sie hin und her zu konvertieren. Uns geling es nicht, aufgrund von gelinde gesagt technischen limitierungen.

Aufgabe 2

Optimaler Ablauf

- Man führt das Tool aus mit der Klasse GenerateNewRoomWithDeepSynth.java und dem argument „-roomtype **x**“ wo **x** ist bedroom, office oder living. Z.B. „-roomtype office“
- Dies baut ein Kommando zusammen, zB
    - python batch_synth.py --save-dir aufgabe_2_tmp_data --data-dir bedroom --model-dir res_1_bedroom --start 0 --end 1 --rotation-epoch 180
- Dies führt das deepsynth-Python-Skript mit JEP aus und wartet, bis es zu Ende läuft.
- Es werden weitere SUNCG Objekte eingefügt, solange es der continue_predictor erlaubt.
- Eines der Ergebnisse wird mit dem Stichwort Final versehen z.B. **0_0_8_final.json**

- Das Importer-tool (GenerateNewRoomWithDeepSynth.java) sucht nach einer Datei mit so einem Stichwort und liest **0_0_8_final.json** dann ein, als Interne Repräsentation der Szene. Es ersetzt alle Objekte mit ihren ShapeNet Pendants.
- Nach dessen Durchlauf erhalte man eine Shapenet XML Datei **0_0_8_final.xml**, die standardmäßig mit der internen Speicherfunktion produziert wird

Probleme:

Man brauche im Aktuellen Stand zugleich zugriff auf CUDA Grafikkarten und auf IntelliJ. Auf Rawindra kann man IntelliJ nicht installieren, da nur Kommandozeile und Lokal hat keiner von unserer Gruppe CUDA zu verfügung. D.h. wir müssen den Vorgang aufspalten und den CUDA-Teil Lokal nur simulieren. Wir laufen deep-synth manuell und fügen dann das Ergebnis in in %importer%/deep-synth/aufgabe_2_tmp_data/**0_0_8_final.json** und erst dann führen wir das Tool (GenerateNewRoomWithDeepSynth.java) aus mit der Flagge -demo, sodass es mit CUDA nicht gerechnet wird und mit der Flagge -nodel, sodass es die Eingabe Für unser tool nicht gelöscht wird, da es erwartet wird, dass diese von deep-synth angelegt wird und deshalb wird sie ansonsten Sauberkeitshalber gelöscht.

Die Einrichtung von Deep-Synth hat auch die veränderung von create_data gefordert und auch eine Versionsveränderung gewisser deprecateden Module. Daher Ist mit drinne in Github repo auf https://github.com/janlycka/suncg_shapenet_tool das Verzeichnis Namens deep-synth-stand-april-21, wo alle Dateien gespeichert sind, zum Zeitpunkt unserer Abgabe. Von Interesse sind create_data.py und requirements2018.txt. Dies wurde in einer Issue angegangen https://github.com/brownvc/deep-synth/issues/8 .