Prof. G. Zachmann
T. Hudcovic (hudo@uni-bremen.de)
R. Fischer (r.fischer@uni-bremen.de)

Winter Semester 2024/25

Assignment on Virtual Reality and Physically-Based-Simulation - Sheet 4

Due Date January 13, 2024

In this assignment we will look at different aspects of interaction in virtual environments. As a specific example, we will deal with the interaction with interlocking blocks (a well-known brand of such interlocking blocks is Lego).

On our website you will find an Unreal project that works with Unreal version 5.3 (possibly also 5.4). Many functions like merging multiple interlocking blocks to one actor are already implemented.

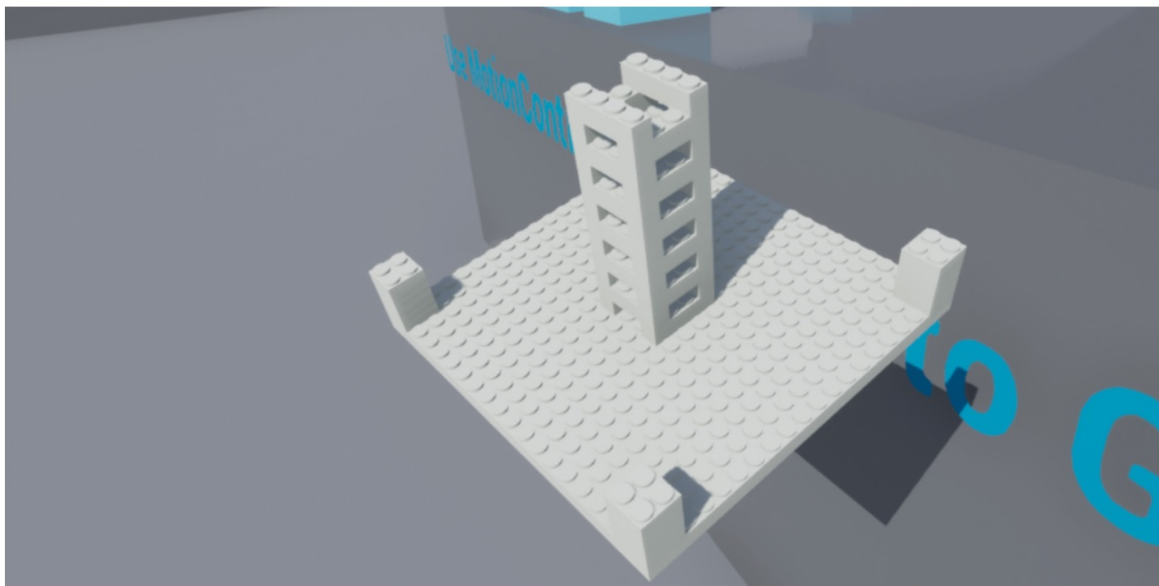## Exercise 1 (Analysis, *4 Credits*)

Figure 1: An example of assembled blocks in the given Unreal project.

Start the project in VR mode and walk to the table on which the interlocking blocks are lying. Try to interact with these blocks and put them together so that you get the presented example (see Fig.1).

(a) Identify the **interaction metaphor** and the **interaction task** for interaction with these blocks.

(b) Describe the current implementation of the interaction metaphor. Pay attention to details, e.g. when the terminal blocks are merged (bullet points are sufficient).

(c) What are the advantages and disadvantages of the present implementation in your opinion? Describe them shortly (bullet points are sufficient).

(d) What other interaction metaphors might be considered for this interaction task (or a subtask)? Name at least two more and describe each in no more than three sentences.

Exercise 2 (Plan a new interaction metaphor, *5 Credits*)

This assignment is about considering the possibilities of interaction metaphors in order to put together interlocking blocks and finally develop a concrete, new interaction metaphor on paper. In a certain part, the interaction metaphor may be similar to the current one, however, there should be significant differences. Also note that the interaction metaphor should be as simple and precise to use as possible.

(a) Draw a nearly complete taxonomy (or decomposition) for the interaction task.
(b) Describe the interaction metaphor you plan to implement and list in bullet points the advantages and disadvantages of your planned interaction metaphor.
(c) Basically, it makes sense to let the user know which object is selected or how objects are changed before performing an action. Consider at which step of your designed interaction you want to use object highlighting. Highlighting should be used at at least one stage of your interaction metaphor. Briefly describe when you want to highlight which components

Exercise 3 (Implement the new interaction metaphor, *15 Credits*)

Implement the planned interaction metaphor and the highlighting of components.

Some general hints:

The framework essentially consists of the C++ classes BlockBaseActor and BlockBaseComponent which represents the interlocking blocks (can be found in C++Classes/Blocks/Base). There are currently the differ- ent types of blocks which inherits from the BlockBaseActor and are called Block1x4Actor, Block20x20Actor and Block2x2Actor (can be found in C++Classes/Blocks/Blocks) - if you want to add some more blocks into the level, use them!

An instance of a BlockBaseComponent describes a single interlocking block in the world and inherits from the UStaticMeshComponent. Such a UStaticMeshComponent inherits also from the known USceneComponent - this means that such a block can be positioned, rotated and scaled relative to the actor. A BlockBaseCom- ponent needs to be attached to a BlockBaseActor to be visible in the level.

The BlockBaseActor is a movable and tangible actor, which can have several merged BlockBaseComponent. This actor implements the following methods:

1. the mergeTo function, which requires another BlockBaseActor and is meant to merge this actor to the given one. Internally, it copies all BlockBaseComponent's to the given actor and deletes itself. Before merging, it checks, whether the interlocking bricks can be merged based on their location and orientation. In case that no merging is possible, the function returns false. (for your own safety, better don't modify this method, only use it).
2. the overwritten OnOverlapBegin and OnOverlapEnd functions, which are called as soon as a component of the actor roughly overlaps with another component (you can change this method).
3. the overwritten PickupImplementation and DropImplementation functions that are called when the user reaches for a BlockBaseActor (you can change this method).

Exercise 4 (Bonus, *max 5 Credits*)

Consider and implement a possibility to detach terminal blocks again (if you don't use git, better keep a working copy of your solution of the previous exercises before you start this exercise)