Prof. G. Zachmann
J.Rosskamp (j.rosskamp@uni-bremen.de)
T.Hudcovic

University of Bremen
School of Computer Science
CGVR Group
December 11, 2023

Winter Semester 2023/24

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

Due Date 08.01.2024, 23:59 Uhr

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

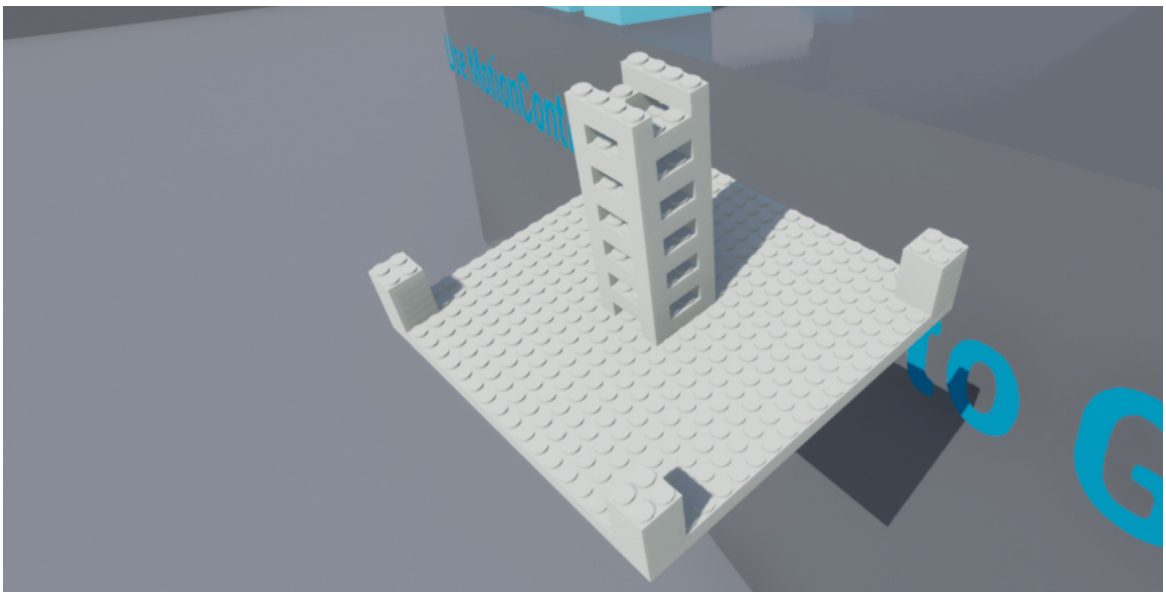

Figure 1: An example of assembled blocks in the given Unreal project.

On our website you will find an Unreal project[1] that works with Unreal versions 5.3. Many functions like merging multiple interlocking blocks to one actor are already implemented.

Exercise 1 (Analysis, *4 Credits*)

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).

[1] https://cgvr.informatik.uni-bremen.de/teaching/vr/uebungen/Blocks.zip

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 different 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 BlockBaseComponent 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:*

- *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).*
- *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).*
- *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*)

This time you can choose from three bonus tasks. However, it is not possible to get more than 5 points on all bonus tasks.

a) Implement a way to change the color for individual instances of the interlocking blocks in the editor (max. 1 pts).

b) Design and implement another type of interlocking block, which is not symmetrical and has e.g. an L or T shape. See how the other types are implemented (e.g. Block1x4Actor) and use it as a reference *(To solve this task, you must also model the desired stone. You have to keep in mind that the stone is rasterized and that individual docking points must have a size of 1cm x 1cm x 0.5cm. Make sure that* ***the center*** *of the docking point of the voxel in the internal data structure at (0,0,0) is at the origin of the model. If necessary, export one of the 3D models of the given block types and see how it behaves there)* (max. 3 pts).

c) 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) (max. 5 pts).