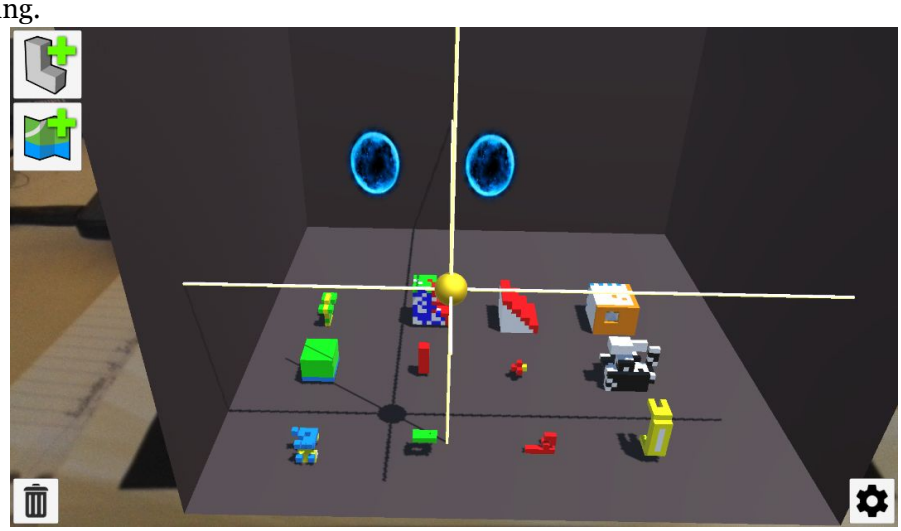
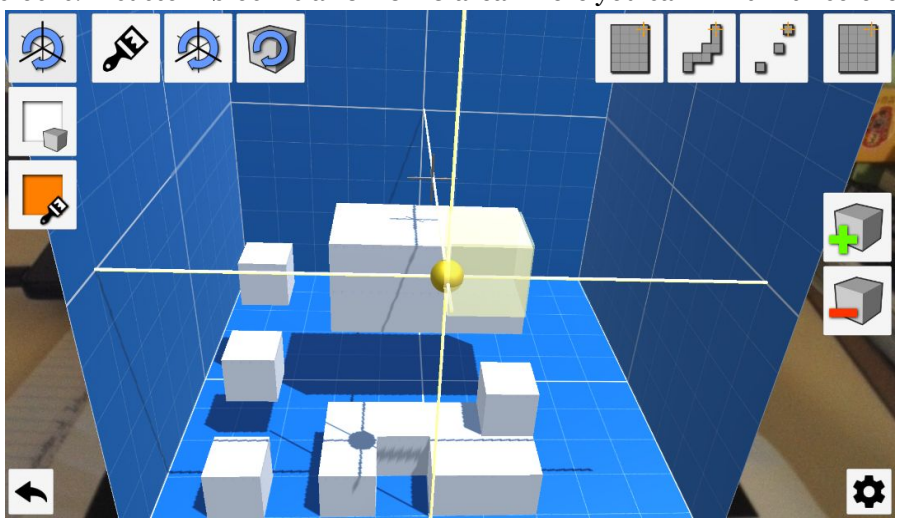
Project title: ARCraft
Ruoqi HE, Xinyi YANG, Yali ZHU, Zhiling PENG

Project goal:
This project aims at exploring the application of augmented reality in the creative process on mobile devices. We choose to create a 3D modeling / scene building application which takes full advantage of the 3D perception provided by AR. By using Multimarker tracking, we can interact with a virtual workspace on the desk, allowing us to create custom 3D objects and scenes. We represent the scene with discretized blocks, which give it a cartoon-like appearance. To interact with the scene, we use a 3D cursor with a fixed related position to the camera for pointing tasks such as adding and removing blocks in space. Touch interaction is also used for selection, painting and rotation. The application also features a two-level scene building pipeline, where we first build our custom block units and then place them in the larger scenes. Modification to the custom blocks is directly reflected in the scenes.

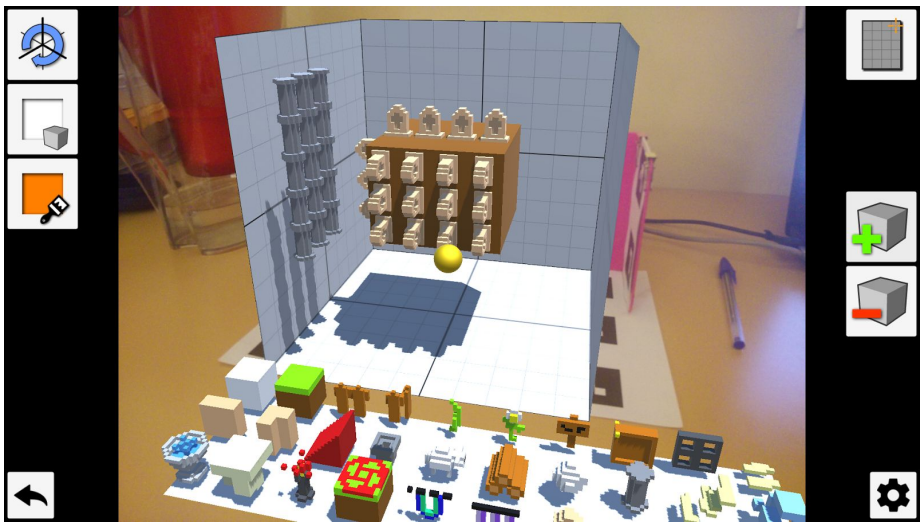
Interface:
1. Warehouse - The first thing to show when starting the application. It stores all your custom objects (blocks) and scenes (portals). From here you can create new custom blocks, create new scenes, edit existing custom blocks/scenes by touching them, or delete everything.



2. Workspace block editing mode (Blueprint background) - This is where you create your custom blocks. A custom block is a 10x10x10 area where you can fill it with colored cubes.



3. Workspace world editing mode (White paper background) - This is where you build your scenes. A scene is also a 10x10x10 area and here you can put your custom blocks as well as basic colored cubes.



List of functionalities:

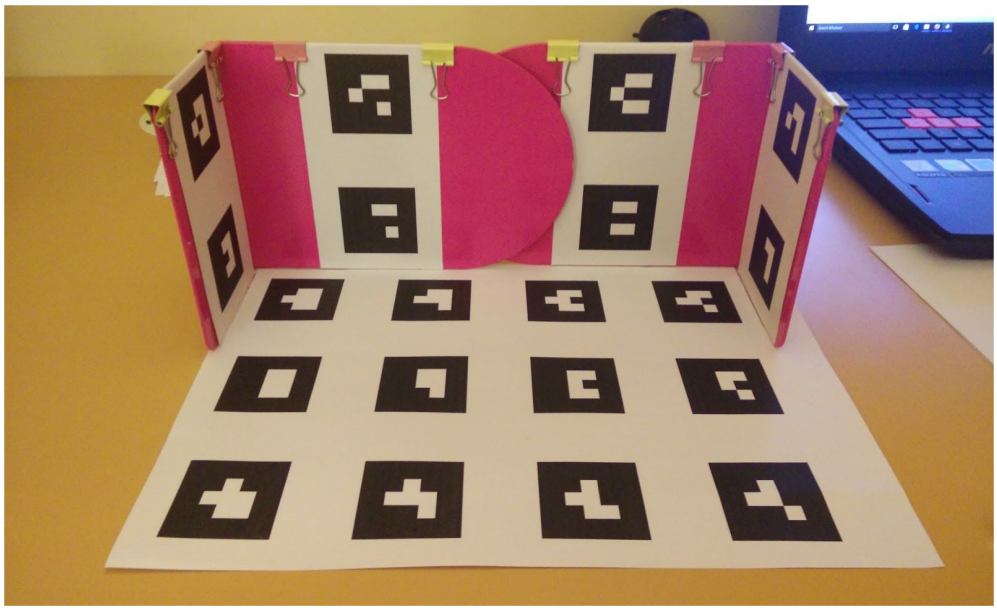
1. Visualize the 3D cursor position with alignment lines to the grids on the workspace walls. The walls are only rendered from the inside, causing no occlusion to the working object.
2. Create and delete blocks at 3D cursor position in workspace. Three modes are supported:
    a. Discrete mode - Add/remove blocks one at a time
    b. Continuous mode - Hold the button to continuously add/remove blocks
    c. Box mode - Drag a box and add/remove blocks in the whole area
3. Choosing color for blocks or for brush to paint the blocks with a virtual 3D color picker. From the color ring you can select one of the 12 color hues, and each you have 5 (value) x 5 (saturation) color options to choose from, displayed as colored cube arrays in the middle.
4. Paint the block by drawing on the screen with your finger when in paint mode (selected from the up-left corner). Uses raycasting to paint the first block hit.
5. Rotate the whole workspace via flick gestures to change view perspective, when in world rotation mode (selected from the up-left corner). We rotate along one of the three axis (x, y, z) which best matches the finger movement.
6. Rotate the selected custom block via flick gestures in order to place it in different poses, when in world editing mode + block rotation mode (selected from the up-left corner). We use the same algorithm as above.
7. Create custom blocks and save them in the warehouse (block editing mode).
8. Create scenes with basic colored cubes and custom blocks available in the warehouse (world editing mode). The block to use can be selected from a block palette in front of the workspace, by tapping on it.
9. Save all your work (custom blocks & scenes) in the local file system. Automatically load them when starting the application.

Technical difficulties:

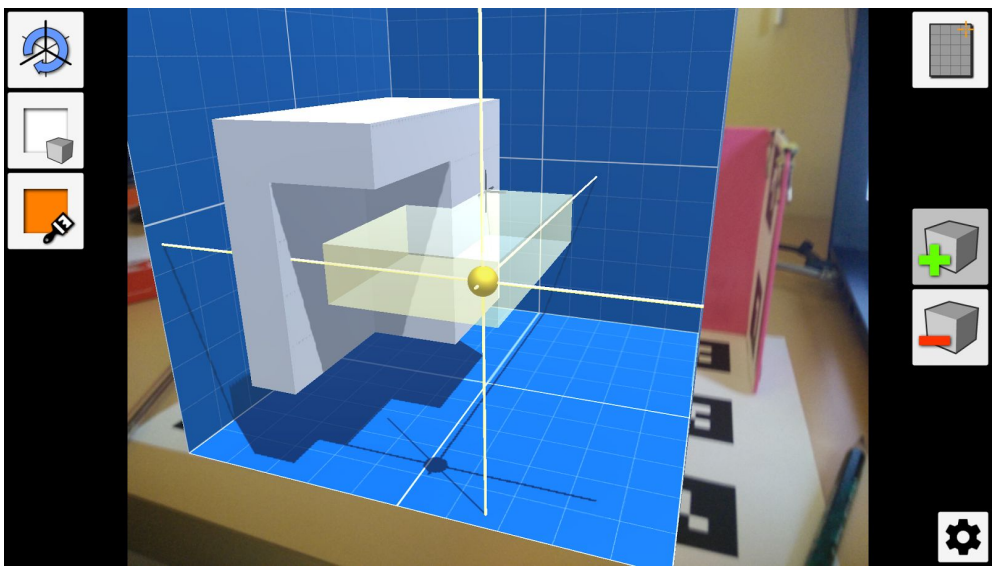
1. ARToolkit's built-in multimarker file contains errors. It took us some time to find it out.
2. On Android the ARToolkit uses its default resolution 320x240 which is very poor. We figured out a way to modify it via ARToolkit's camera preferences. However, we had to modify the ARController script to bring up the debug UI containing the

setting menu via button presses.

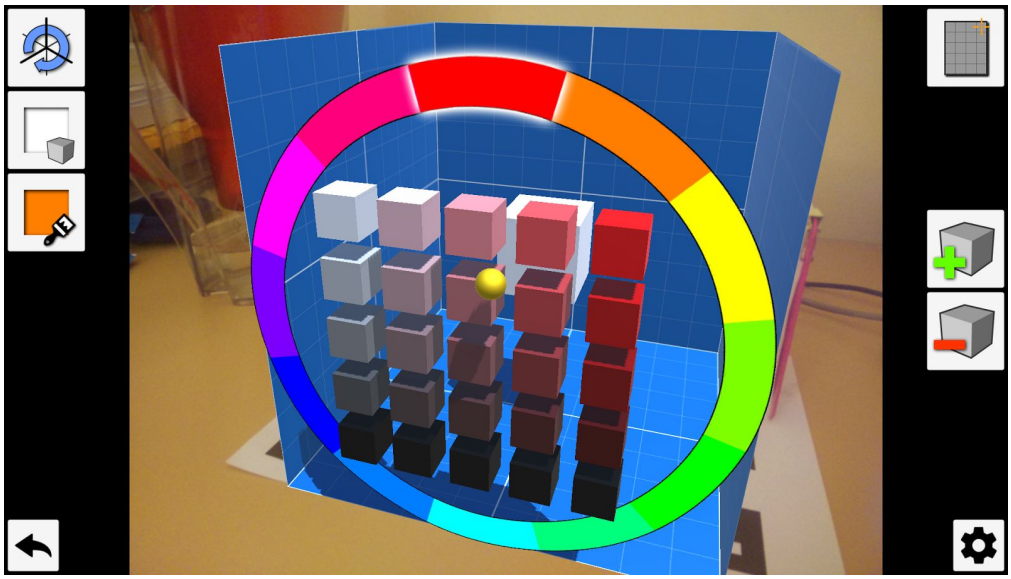
3. When using 16:9 resolution, the detection will sometimes be unstable for unknown reason. So we stick to 4:3 resolutions.
4. As the ARCamera adjust its position in LateUpdate phase, the position provided by the 3D Cursor (which stays relatively fixed to the camera) will be incorrect for other scripts that use it during Update phase. Our solution is to create a CustomARCamera class that extends the ARCamera and move its LateUpdate function to the Update phase. We also modified the script execution order in project setting, such that all other scripts (except ARToolkit scripts) will update after the camera's update.
5. We originally wanted to create a system that auto-detects new markers on the desk and adds it into the multimarker array. However as ARToolkit's multimarker definition is located in an external file, we found it difficult to modify it at runtime. We thus stick to fixed multimarker.



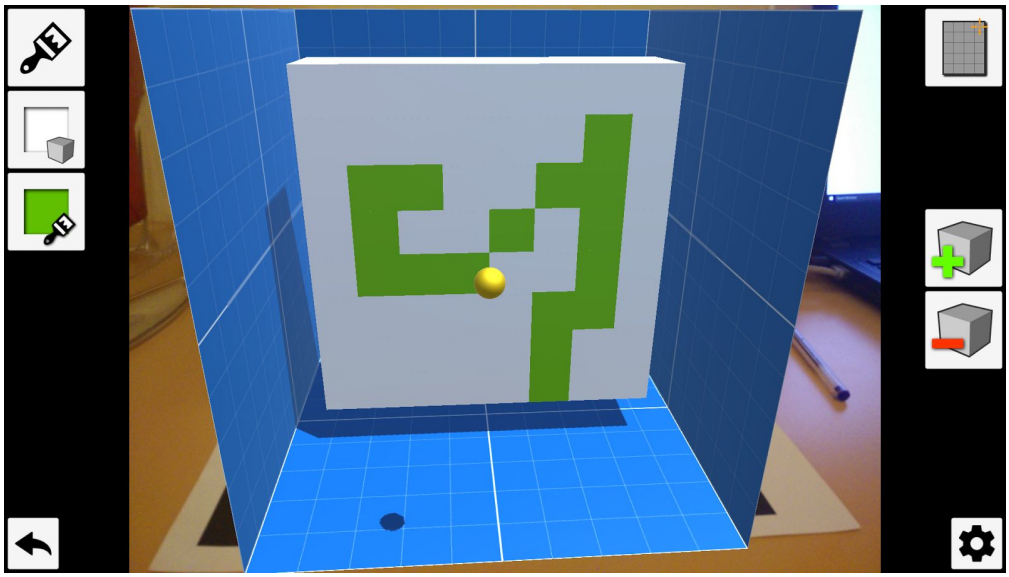
Multi-marker array - The app is still usable with only the base markers



Blocks placing - box mode



The color picker



Brush painting

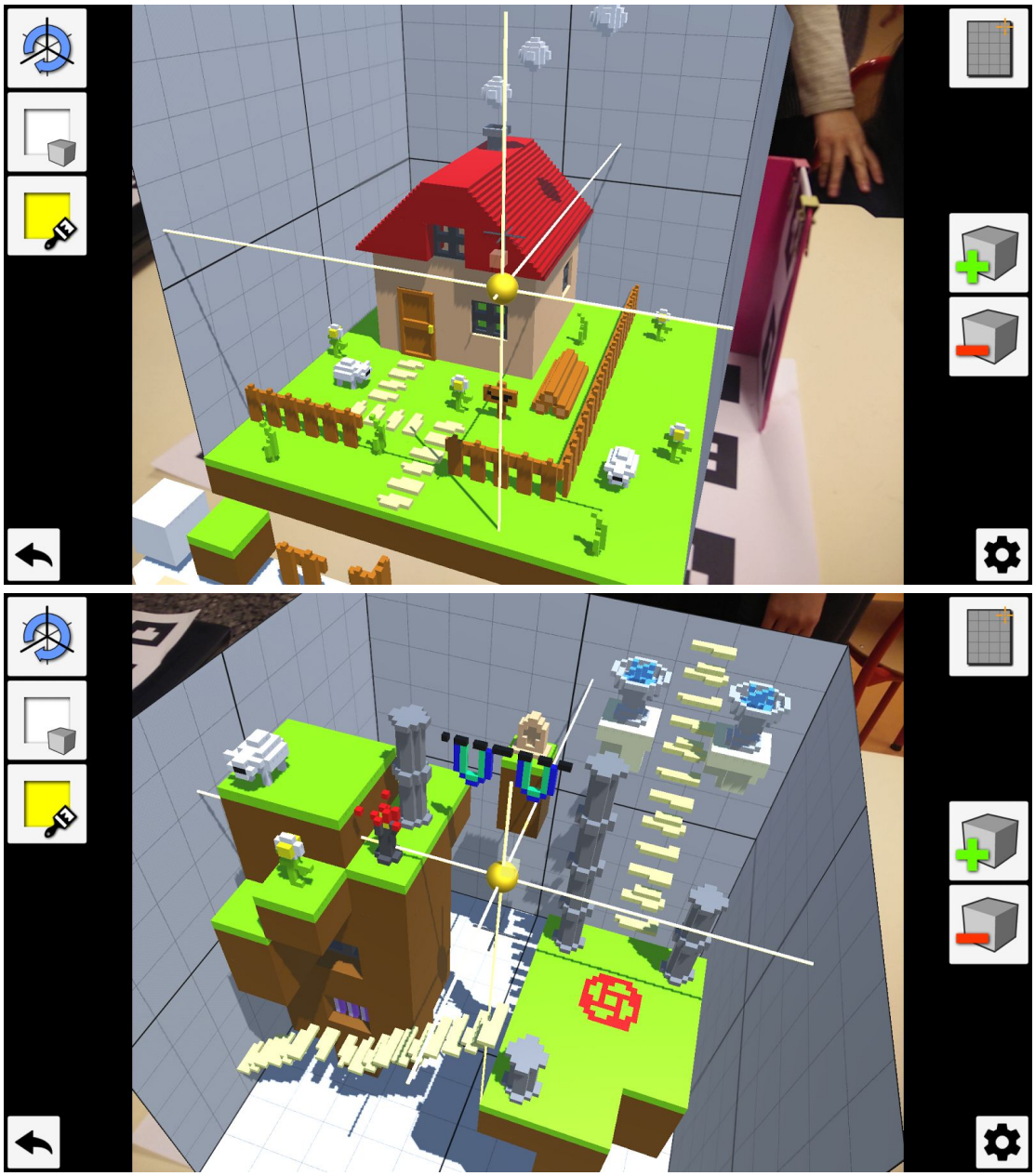


Fig 5. Scenes built by us using ARCraft.