Building a Programming System in and for Virtual Reality

Master’s Project Proposal, Summer Term 2023
Software Architecture Group
Prof. Dr. Robert Hirschfeld

When developing applications in virtual reality, programmers are faced with the challenge of incompatible means of interaction: while developing, they sit at a desk with a keyboard; to test a change, they have to stand up, put on the VR headset, pick up the controllers, and walk to the correct point in their room. If, instead, programmers could modify their applications’ code within virtual reality, they could benefit from significantly faster feedback loops.

In this project, students will explore different aspects around the idea of enabling programmers to work in virtual reality. Potential directions include efficient means to enter text, direct manipulation akin to block-based programming [4], and collaboration between programmers. The project will be based on a research prototype of a self-sustaining virtual reality programming environment [1] that is written in Squeak/Smalltalk [2]. **Students will:**

- **perform rapid prototyping and user testing to identify promising ideas for interactions,**
- **implement and integrate their prototypes into a coherent programming environment,**
- **verify the assumptions established during prototyping in user tests.**

Background: Programming for Virtual Reality

Virtual reality offers a unique experience for users: One where they are highly immersed in the digital world, with the ability to interact with objects as they would in real life. As such, when developing for virtual reality, it is important for programmers and designers to meet the expectations of users concerning interactions, as a mismatch can easily confuse users and leave them unable to interact with the system at all. One important means for programmers and designers to identify when the interactions they create have a natural feel is through feedback. By trying out various thresholds, constants, or even different code paths for interactions, programmers and designers can match their experience to their own expectations to quickly identify promising directions. However, when programmers and designers are constantly forced to move between a development setup and a testing setup, feedback loops are significantly longer and development becomes frustrating.

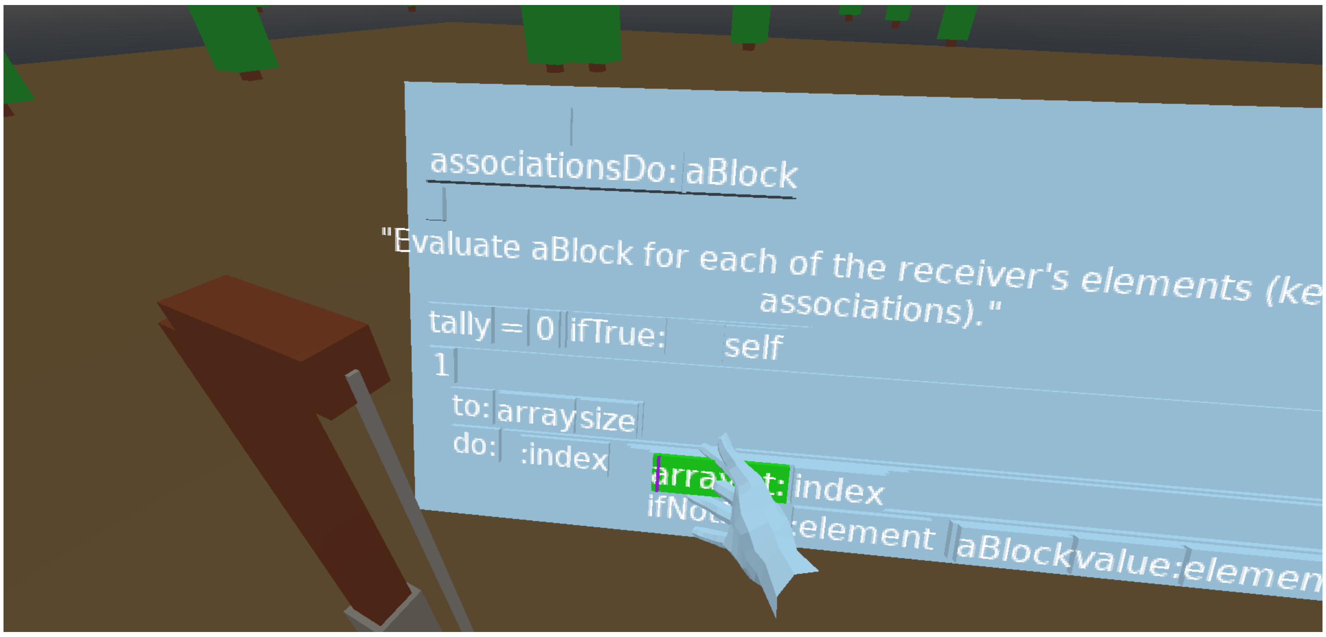
In the context of software development outside VR, some programming systems offer means for programming that drastically reduces the distance between programming and testing through live programming [3]: the program and the development environment run in the same system and thus allow programmers to relate any live objects to the code they are writing.

Approach: A Self-sustaining Programming System in Virtual Reality

In the process of reaching our goal of a programming system in virtual reality, a number of interesting directions for next steps have surfaced. The team will be able to select and direct these and other concerns that arise in their own work.

Efficient and ergonomic text input is still an unsolved topic. Possible directions include gesture-based mechanisms, speech-to-text, or even a system that uses non-textual identifiers.

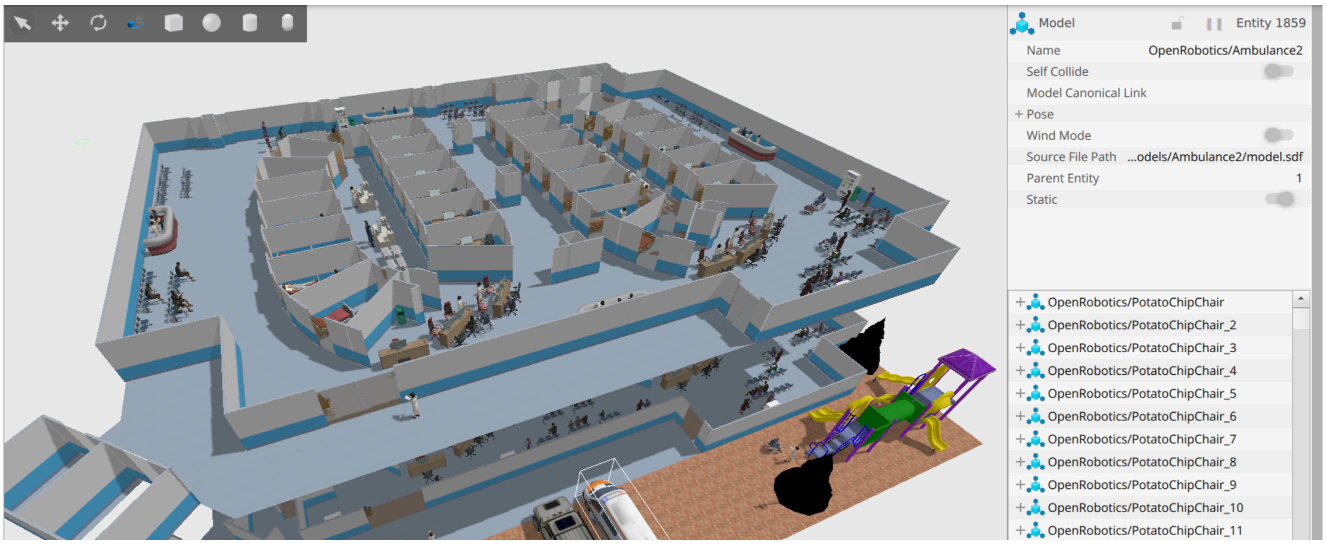
As another major point, programming often is a collaborative activity, where multiple people work together at the same time. Here, the team could explore different means to assist collaboration in a virtual reality system: users could share a single virtual space, potentially while only one user has write access for ease of implementation. Alternatively, asymmetric collaboration with one user in virtual reality and another user at a computer with write access to that virtual system's running code would bridge the gap of text input efficiency, with editing and testing spread across two roles.



```
associationsDo: aBlock
"Evaluate aBlock for each of the receiver's elements (ke
                    associations)."
tally = 0 ifTrue:    self
1
  to: array size
  do: :index
             arra  t: index
             ifNo    :element  aBlock value: elemen
```

Screenshot of our prototypical VR IDE. Visible is a piece of code next to a bow-and-arrow application.

A challenge that came up during our exploration concerns the many modalities of different applications co-existing in the virtual programming system: Unlike a 2D user interface, where users select the context they want to interact with by moving the mouse cursor to the respective buttons, lists, etc., applications in our prototypical implementation tended to demand exclusive ownership of the the user's hands, as interfaces close to the user tended to be more ergonomic. The challenge is thus to design a framework for multiple applications to co-exist in a way that the user can interact with each without conflict, while still allowing those applications to choose the most ergonomic interactions for their use cases.



Model	Entity 1859
Name	OpenRobotics/Ambulance2
Self Collide	
Model Canonical Link	
+ Pose	
Wind Mode	
Source File Path	...odels/Ambulance2/model.sdf
Parent Entity	1
Static	

+ OpenRobotics/PotatoChipChair
+ OpenRobotics/PotatoChipChair_2
+ OpenRobotics/PotatoChipChair_3
+ OpenRobotics/PotatoChipChair_4
+ OpenRobotics/PotatoChipChair_5
+ OpenRobotics/PotatoChipChair_6
+ OpenRobotics/PotatoChipChair_7
+ OpenRobotics/PotatoChipChair_8
+ OpenRobotics/PotatoChipChair_9
+ OpenRobotics/PotatoChipChair_10
+ OpenRobotics/PotatoChipChair_11

3D environment in a simulation. CC BY 3.0 US, Open Robotics, https://gazebosim.org/showcase

Further, the team could explore the opportunities for visualization and direct interaction that virtual reality affords. For example, a system visualizing digital twins or simulated replicas of robots interacting with one another could be integrated, where aspects such as lossy communication between the robots are visualized in the virtual space. By integrating with the programming system, users can immediately tweak aspects of the simulation and observe the impact live.

Lastly, most user interface systems have evolved to support some form of “developer tools” that allows the live inspection of state of the user interface. Exploring how this can be communicated and integrated in the less clear-cut domain of 3D objects and visual effects (compared to rectangular, typically axis-aligned 2D boxes, i.e., buttons/windows) is another challenge.

Other practices and values

Within this project, we want to follow certain values and practices:

- **Explorable Implementation**: The majority of the solution will be implemented in Smalltalk to help programmers follow an exploratory workflow and interactively debug and resolve issues with Squeak’s existing toolset.
- **Self-directed Work:** The team is expected to identify and follow up on interesting directions around the general theme provided by the proposal. The team will receive feedback on their solutions and possible other directions they may want to explore; they will be expected to incorporate that feedback in their design process (but not necessarily in their implementation).
- **User Testing:** On a regular basis, parts of the proposed solution need to be prepared and packaged as contributions to the main branch of our VR programming environment. Regular feedback and testing with users once a stable base has been established needs to be strongly considered.

Organization

A group of three to five (3-5) students may participate in the project. Organization and tasks will be explored and mainly determined by the project participants. The project will be carried out at the Hasso Plattner Institute in Potsdam or virtually if necessary. Project participants are expected to communicate with the advisors on a regular basis. Communication tools include but are not limited to video chat tools, GitHub issues, projects, a wiki, and e-mail. Participants will regularly present their work to the chair to report progress and to obtain feedback. Communication is likely to be conducted in German or English. Expected results include a working software prototype accompanied by appropriate documentation including its architecture, design decisions, and framework and API usage.

Contact

Prof. Dr. Robert Hirschfeld, Tom Beckmann
Software Architecture Group, HPI, Potsdam
http://www.hpi.uni-potsdam.de/swa, {first.last}@hpi.uni-potsdam.de

[1] Leonard Geier, Clemens Tiedt, Tom Beckmann, Marcel Taeumel, and Robert Hirschfeld. 2022. Toward a VR-Native Live Programming Environment. PAINT 2022. https://doi.org/10.1145/3563836.3568725
[2] D. H. H. Ingalls, T. Kaehler, J. H. Maloney, S. Wallace, and A. C. Kay, “Back to the Future: The Story of Squeak, a Practical Smalltalk Written in Itself,” OOPSLA, 1997, pp. 318–326.
[3] Patrick Rein, Stefan Ramson, Jens Lincke, Robert Hirschfeld, and Tobias Pape. 2019. Exploratory and Live, Programming and Coding - A Literature Study Comparing Perspectives on Liveness. The Art, Science, and Engineering of Programming 3, 1 (2019), 1. https://doi.org/10.22152/programming-journal.org/2019/3/1
[4] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, and et al. 2009. Scratch: Programming for All. Commun. ACM 52, 11 (Nov. 2009), 60–67. https://doi.org/10.1145/1592761.1592779