Ghost in The Paper: Player Reflex Testing with Computational Paper Prototypes

Tom Beckmann
Hasso Plattner Institute
University of Potsdam
tom.beckmann@[*]

Eva Krebs
Hasso Plattner Institute
University of Potsdam
eva.krebs@[*]

Leonard Geier
Hasso Plattner Institute
University of Potsdam
leonard.geier@[*]

Lukas Böhme
Hasso Plattner Institute
University of Potsdam
lukas.boehme@[*]

Stefan Ramson
Hasso Plattner Institute
University of Potsdam
stefan.ramson@[*]

Robert Hirschfeld
Hasso Plattner Institute
University of Potsdam
robert.hirschfeld@[*]

Abstract
Paper prototyping is an effective strategy for digital game designers to explore a design space. However, a large area of the design space is out of reach for paper: prototypes that need to react to player reflexes are challenging to realize. Many game genres use player reflexes as an integral part of their concept.

Through a set of examples, we analyze factors that contribute to the effectiveness and spontaneous spirit of paper prototyping. We then propose digital tool support designed to broaden the scope of paper prototyping, by introducing computation to paper arrangement, while maintaining its effectiveness and spontaneous spirit. We present and discuss the design of an explorative study to evaluate the concept.

1. Introduction
Game designers are faced with the challenge of obtaining feedback on their concepts as quickly as possible (Schell, 2019). As they typically work to create the elusive quality of "fun", it is difficult for them to gain insights through any other means than observing players of their game or playing it themselves.

Consequently, game designers frequently create prototypes that reveal an insight about their concept with as little time invested as possible. To this end, paper prototyping, or more generally low-fidelity prototyping, is a popular method that allows designers to quickly materialize a playable concept, even improvising more elements or rules on the fly. Contrarily, in most digital game prototyping methods a designer first needs to laboriously formalize the rules of the game in a way that a computer can execute them. While a designer may prepare a set of parameters to tweak quickly in their digital game, free improvisation is turned nigh impossible, even in settings with fast edit-compile-run cycles like live programming systems.

However, there are certain questions about a concept that are difficult to answer through a paper prototype. Games appeal through a variety of factors, such as engaging puzzles, skill challenges, or social communities. Concepts that seek to exercise the players' reflexes will likely benefit from a digital prototype, as a computer can respond to a player's actions within milliseconds, whereas an interpretation of another human directing a paper prototype will likely take seconds and be subject to significant imprecision.

In this paper, we seek to find a way to bring the benefits of paper prototyping to concepts that want to test player reflexes. To this end, we constrained a digital drawing tool to closely resemble paper prototyping. We then added the possibility for a designer to express rules within that tool that are automatically simulated by the computer. With this setup, we present insights from an exploratory user study pilot that seeks to answer two research questions:

- How does our digital counterpart compare to analog paper prototyping in terms of the designer's and the player's experience?

[*]hpi.uni-potsdam.de

- How well does our digital paper prototyping tool allow for prototyping game concepts that rely on player reflexes?

In the following, we will present insights from several paper prototyping sessions we observed and drew conclusions from for our digital tool (Section 2). We will then briefly describe how we mapped those insights to a digital tool (Section 3). Next, we describe our setup and insights gained from the pilot user study (Section 4) and discuss those (Section 5), before considering related work (Section 6) and concluding the paper.

2. Paper Prototyping

To guide the design of our digital tool for paper prototyping, we first observed two colleagues while they collaboratively designed a paper prototype. In the session, the designers conceptualized and started drafting a single-player card game where the player draws cards from a stack and maintains a hitpoint and defense counter. We interrupted their prototyping session after about one hour. We took notes focused on the activities and actions they performed during prototyping. While activities describe the operation on a high-level, actions describe how the designers physically perform activities.

2.1. Activities

During the prototyping process, multiple activities freely interleave. Designers may spontaneously interrupt one activity, perform another one, and return within seconds. Based on the actions we observed designers perform, we derive three activities: discussion, asset creation, and playtesting.

Discussion At the very start, whenever the next step appeared unclear, or when a new insight surfaced, the designers started brainstorming and discussing ideas. These could involve rules for the game, the look and feel of game elements, or balancing decisions. Often, the discussion served to align ideas or conclusions between the designers.

Asset Creation Here, the designers created pieces for their playable prototype. In the observed instance, this involved paper cut-outs, blank cards, as well as tokens from a board game that the designers found in the room.

Playtesting Once enough assets were available, the designers began playtesting the prototype. Notably, the designers would frequently interrupt playtesting and return to discussion or asset creation. This occurred when a rule appeared to be missing or ambiguous but also when a designer believed to have identified potential for an even better rule. Often, one designer would be playing while the other was observing or creating new assets to quickly inject into the playing session.

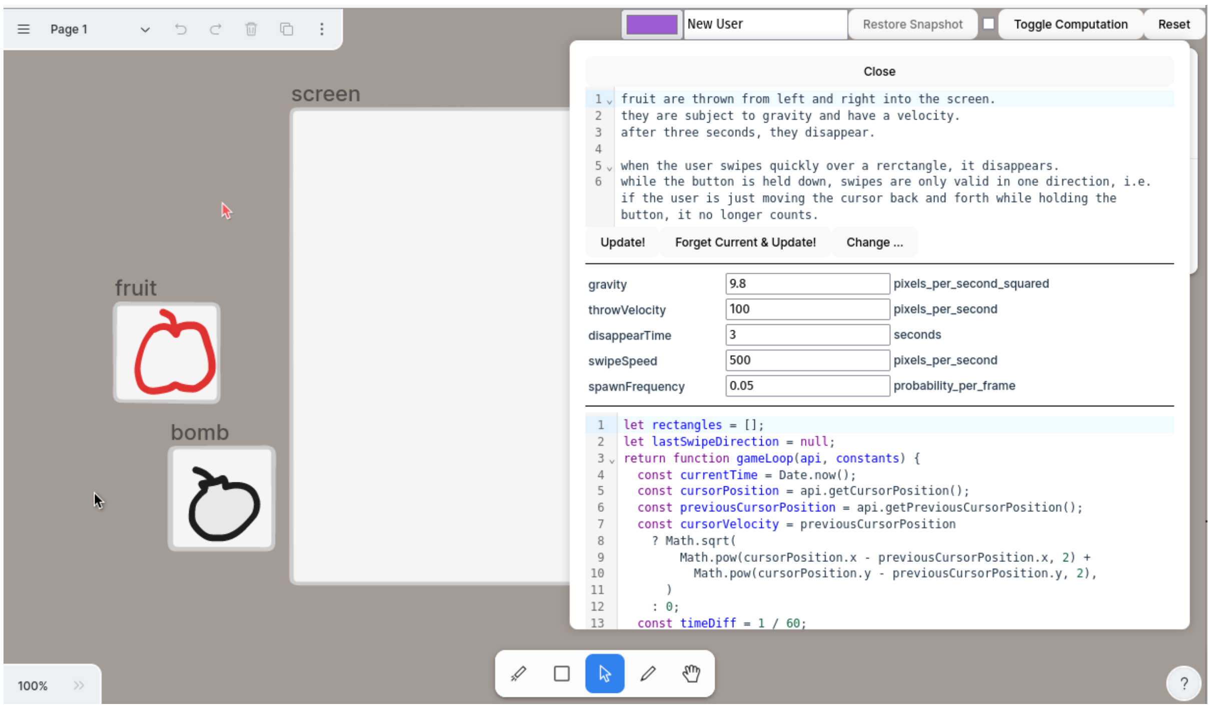
2.2. Operationalization of Activities

During the three activities, the designers performed a variety of actions.

During discussions, the designers made use of a whiteboard. They wrote text on it, drew shapes, circled drawn elements, or connected them with arrows. The relative placement of text and shapes was used to signify relationships.

The designers verbally added context to ad-hoc syntax, such as dotted lines to signify an implicit effect between two elements, or arrows to demonstrate movement. They proposed rules verbally, named challenges, and frequently referenced mechanics from other games as a shortcut for explanations. When discussions interrupted playtesting, the designers pointed at paper elements, gathered relevant elements for a better overview, or asked for opinions from their colleague.

The actions that occur during asset creation and playtesting are of special importance to our goal of creating a digital paper prototyping tool. From the actions the designers performed, we derive three categories: **Arranging**, **Describing**, and **Editing**.



```
fruit are thrown from left and right into the screen.
they are subject to gravity and have a velocity.
after three seconds, they disappear.

when the user swipes quickly over a rerctangle, it disappears.
while the button is held down, swipes are only valid in one direction, i.e.
if the user is just moving the cursor back and forth while holding the
button, it no longer counts.
```

gravity	9.8	pixels_per_second_squared
throwVelocity	100	pixels_per_second
disappearTime	3	seconds
swipeSpeed	500	pixels_per_second
spawnFrequency	0.05	probability_per_frame

```
let rectangles = [];
let lastSwipeDirection = null;
return function gameLoop(api, constants) {
  const currentTime = Date.now();
  const cursorPosition = api.getCursorPosition();
  const previousCursorPosition = api.getPreviousCursorPosition();
  const cursorVelocity = previousCursorPosition
    ? Math.sqrt(
        Math.pow(cursorPosition.x - previousCursorPosition.x, 2) +
          Math.pow(cursorPosition.y - previousCursorPosition.y, 2),
      )
    : 0;
  const timeDiff = 1 / 60;
```

Figure 1 – Screenshot of our digital prototyping tool. Three pieces of paper are on the canvas that the user has named. The computations panel is currently open, showing first the users' instructions, then extracted constants for the generated code, and finally, the code itself in case it needs to be debugged. The second user's cursor is shown in red.

To create game elements, they took paper or cards and drew on them. While drawing, they sometimes also tried to exactly copy previously drawn shapes to another piece of paper. When the designers deemed a change necessary, they simply overwrote a previous shape on the paper, for example increasing a number.

Designers were rotating, stacking, and placing tokens and cards. They shuffled stacks of cards or drew cards from a stack.

Once game elements had been created, designers pointed at elements and named them verbally, for example proclaiming that this is the "draw pile" or "a token worth 5 points". Occasionally, they would also refer by name or point at an element and explain an associated rule or the meaning of an icon they drew. Throughout asset creation, designers also stated rules or revised previously stated ones.

During playtesting, the same **Arranging** actions came into play as the designers rearranged elements to signify changes to the game state.

3. Digital "Paper" Prototyping

The list in Subsection 2.2 provides us with an initial vocabulary of actions that a digital paper prototyping tool needs to support. Fortunately, most of these actions are already supported by digital whiteboards or drawing tools. We chose as a basis tldraw*, an open-source, web-based canvas component. A screenshot of our tool is shown in Figure 1.

In particular, our goal is to create an experience in the digital tool that is as close as possible to analog paper with a single change: designers can make elements move automatically based on a set of rules. As a consequence, we remove functionality that is commonly available in digital tools, such as undo. While there are still significant differences, notably the use of a digital interface as opposed to tangible paper, we hope to thus constrain the effects of other conveniences that the digital medium brings to a minimum, to focus on the single, desired intervention of automatic computation.

*https://github.com/tldraw/tldraw

3.1. Adaptations to tldraw

To make tldraw suitable for our needs, we included a remote collaboration feature, such that two users could work and play on the same canvas using two computers.

In terms of interactions, we removed all built-in tools besides those that we derived from our observations on paper prototyping. To ensure that the desired interactions feel natural, we added a pen/touch hybrid tool:

- By dragging the pen across the canvas, the user creates a piece of paper of variable size.
- By dragging the pen across a piece of paper, the user draws on it.
- By dragging a piece of paper with their finger, the user moves that piece. This way, they can also stack pieces.
- By dragging the canvas with their finger, the user pans the viewport.

We left the default tools for creating rectangles (pieces of paper), drawing, and selecting available, in case users wanted to use a mouse instead of a pen.

Finally, we added two special tools: one tool for pointing at an element and to name or explain it using a text, and a second tool that ignores all inputs but still provides mouse inputs to the computation system, as the interactions during play may overlap with those for editing.

3.2. The Ghost in The Paper

The reason for recreating paper prototyping in the digital space, as described in Section 1, is to allow elements to move automatically, reacting immediately to player input. We ideally want designers to be spontaneous and have quick feedback loops akin to prototyping with paper – ruling out the option of writing code manually. For this purpose, we added a text field that allows users to state rules in natural language, as seen on the right of Figure 1.

As guide for designers, we envision a scenario where they describe to a third, absent designer how elements should behave. That third designer would take the role of computer and execute these steps. As the third designer is not present during creation of the prototype, the rules need to be unambiguous and complete. Notably, not all rules of the prototype need to be spelled out, only those that require the computer to be involved.

Given this semi-formalized description of rules, we had a generative artificial intelligence produce a formal, executable version of the rules in JavaScript. Once JavaScript code has been generated, the user can toggle a play mode, which toggles continuous execution of the generated code for every frame in a game loop. When the play mode is activated, we persist a snapshot of the canvas state and allow designers to revert to that state before any modifications by the computer took place. To avoid any issues from networked play, we constrain the play mode to a single client. The other clients still see the scene evolving but the computational rules, such as checks for mouse position, are not run on their instances.

LLM Configuration We used the *gpt-4o* model in our experiments, with its JSON mode enabled and a temperature of 1. The prompt consisted of multiple system messages:

- Informing the AI that it is meant to create rules for a 2D game.
- The layout of the designers' scene, including the names and explanations that designers have attached to elements.
- The API available to it, auto-generated from a TypeScript file, resembling the **Arranging** actions of our vocabulary.

- The boilerplate for the gameLoop, shown in Listing 1.
- A request to extract and return any constants.
- A set of instructions aimed at preventing misuse of the API we have observed.
- Optionally: the previous code, instructions, and constants the LLM had generated, with the request to match it closely if possible.
- As a user message, the rules they specified.
- The layout of the JSON object to be returned: `{constants: {name: string, default: number, unit: string}[], code: string, instructions: string}`.

We display two buttons for updating the rules: one that includes and one that excludes the previously generated code. In practice including or excluding the previous code did not appear to make a difference, as the newly generated code often deviated significantly from the previously generated one, even for small change requests.

```
// any state you may want to store
// let a = ...
return function gameLoop(api, constants) {
  ...
}
```

Listing 1 – Boilerplate for the Game Loop

3.3. Exemplary Workflow

As an example, let us consider a simple jump-and-run to which we want to introduce a new mechanic: a ghost should follow the player while they are not looking at it. The scenario is drawn from an early experiment we did ourselves.

We begin by drawing pieces of paper for platforms onto the canvas. We then create two pieces of paper on which we draw a player icon and a ghost icon.

One designer is moving the player, the other is moving the ghost, each on their respective device. We observed that it is difficult to realize when the ghost should stop moving, i.e. the player is looking in its direction. An attempt to have the player call out the direction they are looking to turned out to be too unreliable. We also do not want to fully automate the ghost's movement, as we are still unsure about its ideal movement pattern and want to maintain the option to improvise its movement.

Instead, we add a computation rule that displays a piece of paper either on the left or on the right of the screen, depending on the direction the player last moved in. This suffices as a guide for the designer moving the ghost to react quickly.

4. Pilot User Study

We design a user study that answers (1) how the experience of using our digital paper prototyping tool compares to analog paper prototyping, and (2) how well our tool allows prototyping concepts that rely on player reflexes. Here, we are presenting the design of and discussing insights from a pilot study.

4.1. Setup

We recruit participants who state that they have prior experience with paper prototyping. To answer the two questions, we give participants two prototyping prompts: one that we believe would work well for analog paper prototyping and one that we believe would be challenging to prototype. Participants work in teams of two. They work on both prototype prompts once with analog paper and once with our digital tool, for a total of four prototyping sessions.

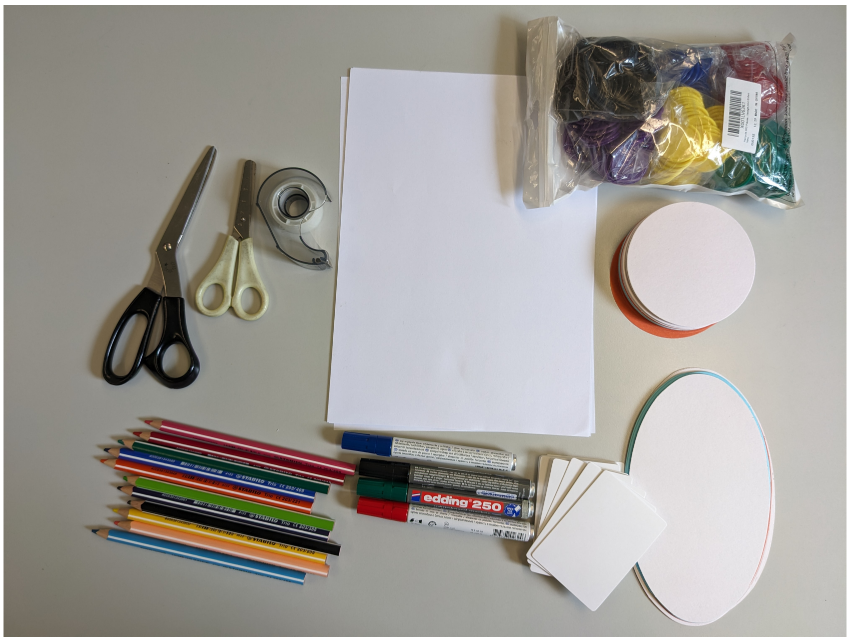


Figure 2 – Materials we provided during the prototyping session for analog prototyping.

For the analog prototypes, we provided a set of materials as shown in figure Figure 2. In addition, we allowed participants to ask for any additional materials, that the instructions would then procure or state that those cannot be used.

Before the first use of our digital tool, participants receive a brief introduction and can try out the tool. We also explain the framing of specifying rules for computations in the setting of recording instructions for a third, currently absent, designer who will later be performing the instructions. In addition, we prime them to consider involvement of computations a "last resort", when it is clear that manual execution will falsify the prototyping insights.

4.2. Prototyping Prompts

For the prompt that should work well with paper, we ask participants to extend the rules of the puzzle video game *Baba Is You*. In *Baba Is You*, the player moves the eponymous Baba on a top-down, rectangular grid. In the grid cells are objects or words. Pushing the words with Baba to form sentences changes the game rules. For instance, building the sentence "Wall is Push" means that Baba can now push wall objects around the grid.

We ask participants to add a new "invert" word to the game that should invert whatever words it is combined with. Participants should investigate two questions through their prototype:

- With what other words of the base game does the "invert" word work well?
- Does the "invert" word allow for interesting puzzles?

For the second prompt, we ask participants to modify the 2D game *Fruit Ninja* that is typically played on a touchscreen. In *Fruit Ninja*, fruits are thrown across the screen and the player tries to slice through these to gain points by swiping across them. Among the fruits are bombs that the player has to avoid.

We ask participants to investigate how the game is best played with a mouse through the question:

- How to best interact with *Fruit Ninja* using a mouse?

4.3. Method

The participants are given the prototyping prompt with a time limit of 30 minutes. While the participants are working on the prompt, we record their interactions and conversations for later analysis. Once the time limit has elapsed, we perform a semi-structured interview with the participants. In particular, we ask the following questions to facilitate answering our research questions concerning the comparison between the mediums and the effectiveness of computations for testing reflexes:

1. What was the most frustrating part about working with the given materials?
2. What worked best in with the given materials?
3. For the digital tool only: What actions you know from analog paper did you want to perform that were not possible?

During early experimentation, we realized that the quality of LLM-generated code was unreliable. Consequently, we opted for a "hybrid" Wizard-of-Oz (Green & Wei-Haas, 1985) mode: participants first generate code, execute it once, and if an error occurs or the behavior is not obvious within the first couple of seconds, we interrupt the session and an instructor repairs the code. We discuss the effect of this approach in Subsection 5.2.

4.4. Results from Pilot Run

As a first pilot for the described study, we executed the setup with two of the co-authors as participants. We briefly describe their approach to answering the prototyping prompt, challenges they encountered, and their answers to the interview.

Baba Is You: Analog The participants first brainstormed interpretations of the "invert" word in combination with other words from the original game. They used a single sheet of paper to document ideas and refer to them later again. To create assets, they made use of a set of multi-colored chips and placed them on a grid that they drew on a sheet of paper, see Figure 3. They drew icons or words on the chips to differentiate them.

During play, the participants quickly realized obvious solutions to their puzzles, rolled back a couple of steps, changed an aspect of the physical level or verbal rules, and tried again. The participants struggled with a consistent interpretation of the "invert" rule throughout the session, as most often they fell back on an interpretation as "not", for which they did not find interesting puzzles.

In the interview, the participants indicated that they kept struggling with the permanence of decisions and that they had to remind themselves that they can always overwrite icons. They found it challenging to quickly move arrangements of objects in their entirety, for instance, a prepared level, without it being damaged in the process. At times, space on the grid they had prepared appeared tight; however, they also noted that they appreciated the constraint to keep levels simple. In parts, it was difficult for them to verify if they were complying with the verbally agreed-upon rules, especially when they were quickly iterating different arrangements of levels.

On the other hand, they also appreciated that the material was not giving any hard constraints – if it was obvious that a sequence of moves was valid, they could move Baba directly to a destination. They expressed that the chips were useful for this prototyping prompt, as they were just heavy enough not to fly around when interacting with the grid and adding new objects by labeling the chips was so quick that they would often do so mid-play.

Baba Is You: Digital After the interview, the participants continued their prototyping session in the digital space. They again started by taking notes, this time on a digital piece of "paper" in our tool. Simultaneously, the other person already started quickly recreating the same tokens by drawing square rectangles and drawing or writing on them.

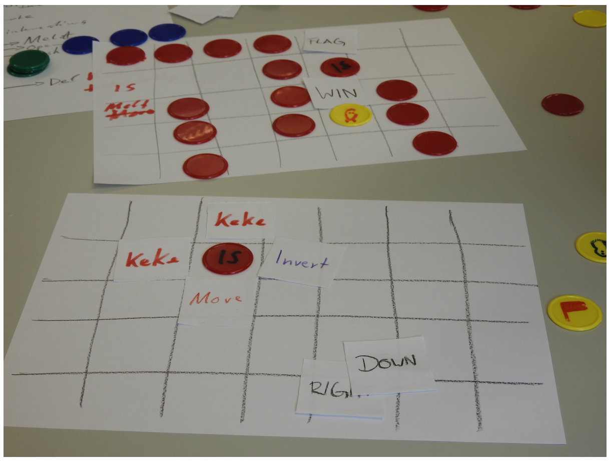


Figure 3 – One analog prototype created for Baba is You.

The mode of asset creation and play were largely the same: participants tried out ideas they deemed promising, adapted aspects mid-play, and discussed insights. When one participant was presenting a new idea, the other participant often tended to look at the presenter's laptop as opposed to their own, even though the partner's cursor positions were shown on both devices.

In the interview, the participants indicated that the pen/touch-mode initially led to one or two wrong inputs but that they quickly adapted. Reliably pointing at small elements using the finger, however, was sometimes challenging. Both participants indicated that there was slightly more friction to creating elements: written text ended up larger and so space was running out more quickly or giving precise inputs required more concentration. To make matters worse, one of the laptops used during the experiment was running a screen recording software, which caused the framerate on that device to be very low. The participants noted two fundamental issues: first, the limited viewports of the two laptops compared to the physical table often made it difficult to see what the other person was working on or even what they were looking at when talking about something specific, making communication more challenging. In particular, shifting the viewport required interactions with the hand on the screen, whereas with analog paper shifting the head or the focal point of the eyes was enough. Second, the participants described an "uncanny valley" of interactions, where the limited set of interactions we constrained our tool to caused some frustration, as they are used to operations like resizing or effortless duplication being available in digital tools. To them, the medium did not evoke the impression of paper enough to remove these expectations.

On the flip side, they praised the effortless creation of paper sheets in the digital tool, as well as the quick selection of the pen color, both of which typically involve reaching across the table or using scissors. The pen/touch-mode also worked very well, after the initial hurdle and aligned well with their expectations from the physical world. In this way, arranging objects, which was the bulk of the session, worked well, too.

In terms of actions they were missing from analog paper, they listed moving multiple things – which was not supported using the pen/touch-mode. They also noted that resizing does work to an extent in the analog world, simply by cutting off parts of a piece of paper. Lastly, rotation was also not well supported in our tool.

Fruit Ninja: Analog The participants again started by sketching ideas on a piece of paper, this time for four different interactions for collecting fruit. They quickly identified as their main challenge to convey a similar feeling to the indirection of using a mouse: an attempt to quickly draw a line between fruit previously drawn on a sheet was deemed too dissimilar; another attempt to tap fruit that the other participant was moving around on the table led to arms getting in each other's way; a third attempt where another person was asked to close their eyes, sit down at the table, and move their finger to collect fruit while another player was "drawing" directions on their back, to simulate the indirection of the mouse, resulted in too high latency and inaccuracy. In a fourth attempt, the participants placed small sheets of paper on the table and two players competed against one another to collect the most, which ended up not feeling comparable to Fruit Ninja. Finally, the closest approximation was putting a table at an incline and throwing chips at it, such that they sled back down while the player had to slice through them; this created a similar feeling but was a better simulation of a touch screen than a mouse.

Consequently, the participants were dissatisfied with their progress, stating that they learned about the prototyping medium but nothing about the prototyping prompt. The major challenges concerned creating continuous, predictable movement, moving enough elements at once to present a challenge to the player, and to move elements without getting into the player's way. However, they also noted that they managed to prototype five distinct ways to approximate the mouse-based interface in the half-hour time span. The chips worked well again here, as they had the right size and weight to be thrown around.

Fruit Ninja: Digital In the final condition, the participants took two digital sheets of paper to sketch their ideas and created a fruit and a bomb asset. They then added computational rules that started out very simple, e.g., "move a shape from left to right on the screen" and gradually increased in complexity, e.g., "slice a rectangle at high speed to remove it". After every generation, the participants made heavy use of the constants to tweak intervals and velocities of elements to find the right feeling. Throughout, the LLM generation failed, meaning that the session had to be interrupted five times for the instructor to fix the code, before the session could resume, which took between 2 and 5 minutes each time. In the end, they prototyped a tap-to-remove interaction and had started tweaking the rules of how a slice-to-remove interaction should work.

In the interview, participants noted that they had wished for copying existing paper or coloring paper. They also noted that the interruptions due to generation failures were jarring and it was difficult for them to pick up where they had left. As the current version of our tool allows only one participant to interact with computations, they had to effectively share one small laptop and one set of controls, even though they saw great opportunity in tweaking constants while the other participant was playing. One aspect that they were missing was the possibility to demonstrate the velocity or the arc of a movement through motion, instead having to express it in text or tweak constants to achieve it.

The participants praised the automatic movement and the quick means to achieve it, allowing them to now have arbitrary numbers of elements moving, without interference from another set of limbs, and with precise and consistent control over movement. As a changed constant instantly made its effect visible, they felt in control when it came to tweaking the behavior.

5. Discussion and Future Work

Here, we briefly discuss insights from the pilot on our study design and on our tool design.

5.1. Study Design

Overall, the prototyping prompts elicited the challenges we had anticipated well: the Baba Is You prompt was quickly realized in both the analog and digital versions and the Fruit Ninja prompt posed a major challenge in analog but was realizable in digital.

As our participants noted the challenge of making sense of the "invert" rule for Baba Is You we are considering instead moving to simply asking participants to design an interesting Baba Is You level with a predetermined set of words. As this was the main aspect that exercised the prototyping medium and that worked well for the participants, we would reduce frustration while keeping relevant insights.

It is still unclear how to best address interruptions when LLM generation fails in the digital medium. One method may be to remove the LLM and instead have a set of higher-level functions prepared that the instructor can invoke. If the set of functions is well adapted to the prompt, it may allow faster cycles.

5.2. Digital Paper

While the digital paper lacked tactile properties and thus led to occasional erroneous inputs, we believe that our approximation of paper prototyping was already mapped well for the prompts we tried. As participants showed frustration that they did not have the functionality available they expected from digital tools, we might consider allowing their use. In particular, they requested scaling and duplicating shapes, which we had removed to better approximate what is possible in the real world. It remains to be seen if the addition of the tools might also change the types of prototypes designers are building in the tool. Whereas they might be inclined to go for a minimal set of elements without a duplicate function, duplication might, perhaps unnecessarily, push them to consider more complex arrangements.

Concerning the computation element, participants were positive in terms of its intention but its execution was challenging as part of the prototyping session. Interestingly, the participants slowed their own progress significantly and introduced further interruptions by incrementally adding rules to supposedly test the system's limits. In practice, the generated code varied so wildly in quality that a previously stable state could not be reliably built upon or even reproduced.

Errors included incorrect syntax (missing delimiters), incorrect use of language constructs (assignment to const variables), hallucinated API methods, writing to internal properties that had been explicitly forbidden by the prompt, or logical errors (modifying a collection while iterating over it, not storing the cursor position from the previous frame to calculate movement). The instructor addressed the issues either by adding further hidden constraints to the prompt and regenerating the code, fixing them manually, and sometimes even asking the participants not to generate code but implementing a change manually when the change was small. In addition, the use of the passed in constants tended to change between code versions, so, even though we kept values and units the same between generated code versions, the behavior changed drastically and participants had to tweak the behavior anew.

An interesting avenue for future work is to explore opting for a full Wizard-of-Oz setup. To realize this, we may consider preparing a set of high-level routines that will likely be of use during the prototyping session. In addition, AI-assisted coding that merely autocompletes as opposed to generating an entire program would give the instructor more control. It remains to be seen if changes can be realized quickly enough this way to keep the participants' attention. Alternatively, prior work has demonstrated the possibility of crowd-sourcing logic for a sketch in real-time (Lasecki et al., 2015). Other work points to the possibility of pre-processing the user sketches to extract structure, which could support the translation process (Li, Cao, Everitt, Dixon, & Landay, 2010).

6. Related Work

Low-fidelity prototypes such as paper prototyping are an essential part of the design process and an essential aid in discovering good designs (Nielsen, 1995). In game design, where according to "the rule of the loop" we want as many iterations as possible to improve our game, paper prototypes are especially relevant for their fast iterations (Schell, 2019). Aside from traditional paper prototypes, other low-fidelity prototypes with a similar essence are also used for game design. This includes physical prototypes, sometimes called "bodystorming", where humans and their interactions are the main focus (Macklin & Sharp, 2016).

Other approaches have combined aspects, or concepts of, paper prototyping with digital or mixed-media tools. For instance, there are several tools such as Miro boards (Chan, Ho, & Tom, 2023) that allow multiple people to collaborate digitally in a way that is similar to using paper and post-it notes. One previous project prepared paper prototypes and manually converted them to digital prototypes in Power Point (Uceta, Dixon, & Resnick, 1998). Even though the general style of the prototype stayed the same and only links between slides were added, the feedback was more concise and focused compared to using traditional paper prototypes because it was easier for testers to suspend their disbelief and not get distracted by the prototype medium.

As our pilot test runs already revealed problems with the limited viewport of a 2D screen, a possibility would be to consider moving to virtual or augmented reality. Previous work used AR to prototype physical or spatial interfaces, e.g. the appearance of elements in cars (Porter, Marner, Smith, Zucco, & Thomas, 2010). There is also a multitude of modeling or drawing tools for virtual and augmented reality, some of which are also targeted at prototyping in particular or rapidly building complex systems in general (Gasques, Johnson, Sharkey, & Weibel, 2019; Kang et al., 2019). The inverse also exists: There are applications that can convert a basis made out of paper to simple VR applications for prototyping purposes (Nebeling & Madier, 2019). In fields like VR where the viewport of the playtester can be controlled, there are also approaches where the user interacts with the virtual environment while a human support team instead of a purely automated environment moves physical objects accordingly (Cheng et al., 2015).

A number of tools and systems facilitate programming interactive experiences like games. Pronto (Krebs, Beckmann, Geier, Ramson, & Hirschfeld, n.d.) and Unreal Engine blueprints (Valcasara, 2015) are both visual approaches to formulating game logic, both of which may speed up iteration time depending on the scenario. Block-based editors such as Scratch (Maloney, Resnick, Rusk, Silverman, & Eastmond, 2010) or Snap (Harvey & Mönig, 2015) provide an environment focused on creating interactive visual experiences that react instantly to changes in programming logic.

Finally, live programming methods are designed to bring faster feedback to programmers (Rein, Ramson, Lincke, Hirschfeld, & Pape, 2019). Some game engines, such as the Godot game engine, make it possible for some changes to be applied live to the running game. In live coding (Blackwell & Collins, 2005), where code is used as a performance art, it is even specifically required to be able to improvise changes over the course of seconds.

7. Conclusion

In this paper, we extracted what may approximate an essence of paper prototyping and translated it to a digital equivalent. In the digital realm, we augmented this baseline with the ability to specify computations to allow designers to create prototypes that react immediately to player reflexes.

In a pilot user study, we found that the digital tool was working similarly to analog paper for the prototyping prompts we tried, but in addition enabled prototyping concepts involving player reflexes. Our attempt to maintain the spontaneous flow of a prototyping session by letting designers formulate natural language and having an LLM generate code turned out to require further work, as the quality of code was highly unreliable.

8. References

Blackwell, A. F., & Collins, N. (2005). The programming language as a musical instrument. In *Proceedings of the 17th annual workshop of the psychology of programming interest group, PPIG 2005, brighton, uk, june 29 - july 1, 2005* (p. 11). Psychology of Programming Interest Group. Retrieved from `https://ppig.org/papers/2005-ppig-17th-blackwell/`

Chan, T. A. C. H., Ho, J. M.-B., & Tom, M. (2023, March). Miro: Promoting collaboration through online whiteboard interaction. *RELC Journal*, 003368822311650. Retrieved from `http://dx.doi.org/10.1177/00336882231165061` doi: 10.1177/00336882231165061

Cheng, L.-P., Roumen, T., Rantzsch, H., Köhler, S., Schmidt, P., Kovacs, R., ... Baudisch, P. (2015). Turkdeck: Physical virtual reality based on people. In *Proceedings of the 28th annual acm symposium on user interface software & technology* (p. 417–426). New York, NY, USA: Association for Computing Machinery. Retrieved from `https://doi.org/10.1145/2807442.2807463` doi: 10.1145/2807442.2807463

Gasques, D., Johnson, J. G., Sharkey, T., & Weibel, N. (2019). What you sketch is what you get: Quick and easy augmented reality prototyping with pintar. In *Extended abstracts of the 2019 chi conference on human factors in computing systems* (p. 1–6). New York, NY, USA: Association for Computing Machinery. Retrieved from `https://doi.org/10.1145/3290607.3312847` doi: 10.1145/3290607.3312847

Green, P., & Wei-Haas, L. (1985, October). The rapid development of user interfaces: Experience with the wizard of oz method. *Proceedings of the Human Factors Society Annual Meeting*, *29*(5), 470–474. Retrieved from `http://dx.doi.org/10.1177/154193128502900515` doi: 10.1177/154193128502900515

Harvey, B., & Mönig, J. (2015, October). Lambda in blocks languages: Lessons learned. In *2015 IEEE blocks and beyond workshop (blocks and beyond)* (p. 35-38). USA: IEEE. doi: 10.1109/BLOCKS.2015.7368997

Kang, S., Norooz, L., Bonsignore, E., Byrne, V., Clegg, T., & Froehlich, J. E. (2019). Prototypar: Prototyping and simulating complex systems with paper craft and augmented reality. In *Proceedings of the 18th acm international conference on interaction design and children* (p. 253–266). New York, NY, USA: Association for Computing Machinery. Retrieved from `https://doi.org/10.1145/3311927.3323135` doi: 10.1145/3311927.3323135

Krebs, E., Beckmann, T., Geier, L., Ramson, S., & Hirschfeld, R. (n.d.). Pronto: Prototyping a prototyping tool for game mechanic prototyping. , *34th Annual Workshop*, 157–168. Retrieved from `https://www.ppig.org/files/2023-PPIG-34th--proceedings.pdf`

Lasecki, W. S., Kim, J., Rafter, N., Sen, O., Bigham, J. P., & Bernstein, M. S. (2015). Apparition: Crowdsourced user interfaces that come to life as you sketch them. In *Proceedings of the 33rd annual acm conference on human factors in computing systems* (p. 1925–1934). New York, NY, USA: Association for Computing Machinery. Retrieved from `https://doi.org/10.1145/2702123.2702565` doi: 10.1145/2702123.2702565

Li, Y., Cao, X., Everitt, K., Dixon, M., & Landay, J. A. (2010). Framewire: a tool for automatically extracting interaction logic from paper prototyping tests. In *Proceedings of the sigchi conference on human factors in computing systems* (p. 503–512). New York, NY, USA: Association for Computing Machinery. Retrieved from `https://doi.org/10.1145/1753326.1753401` doi: 10.1145/1753326.1753401

Macklin, C., & Sharp, J. (2016). *Games, design and play*. Boston, MA: Addison-Wesley Educational.

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010, nov). The scratch programming language and environment. *ACM Trans. Comput. Educ.*, *10*(4). Retrieved from `https://doi.org/10.1145/1868358.1868363` doi: 10.1145/1868358.1868363

Nebeling, M., & Madier, K. (2019). 360proto: Making interactive virtual reality & augmented reality prototypes from paper. In *Proceedings of the 2019 chi conference on human factors in computing systems* (p. 1–13). New York, NY, USA: Association for Computing Machinery. Retrieved from `https://doi.org/10.1145/3290605.3300826` doi: 10.1145/3290605.3300826

Nielsen, J. (1995). Using paper prototypes in home-page design. *IEEE Software*, *12*(4), 88-89. doi: 10.1109/52.391840

Porter, S. R., Marner, M. R., Smith, R. T., Zucco, J. E., & Thomas, B. H. (2010). Validating spatial augmented reality for interactive rapid prototyping. In *2010 ieee international symposium on mixed and augmented reality* (p. 265-266). doi: 10.1109/ISMAR.2010.5643599

Rein, P., Ramson, S., Lincke, J., Hirschfeld, R., & Pape, T. (2019). Exploratory and live, programming and coding - A literature study comparing perspectives on liveness. *Art Sci. Eng. Program.*, *3*(1), 1. Retrieved from `https://doi.org/10.22152/programming-journal.org/2019/3/1` doi: 10.22152/PROGRAMMING-JOURNAL.ORG/2019/3/1

Schell, J. (2019). *The art of game design* (3rd ed.). London, England: CRC Press.

Uceta, F. A., Dixon, M. A., & Resnick, M. L. (1998, October). Adding interactivity to paper prototypes. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *42*(5), 506–510. Retrieved from `http://dx.doi.org/10.1177/154193129804200513` doi: 10.1177/154193129804200513

Valcasara, N. (2015). *Unreal engine game development blueprints*. Packt Publishing Ltd.