FabO: Integrating Fabrication with a Player's Gameplay in Existing Digital Games

Dishita Turakhia
MIT CSAIL
Cambridge, MA, USA
dishita@mit.edu

Harrison Mitchell Allen
MIT CSAIL
Cambridge, MA, USA
hmallen@mit.edu

Kayla DesPortes
New York University
New York City, NY, USA
kayla.desportes@nyu.edu

Stefanie Mueller
MIT CSAIL
Cambridge, MA, USA
stefanie.mueller@mit.edu

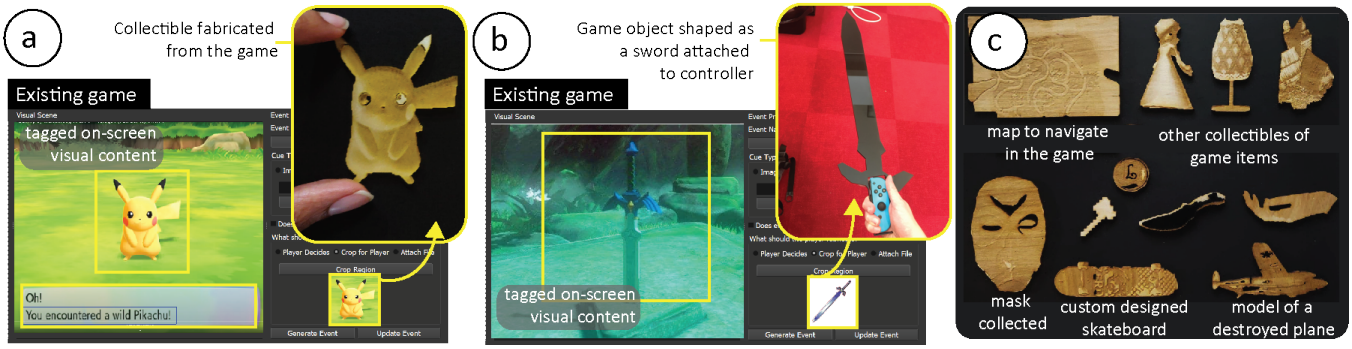


Figure 1: *FabO* allows designers to integrate fabrication of with existing digital games. When players play these integrated games, *FabO* generates fabrication files of objects from their gameplay that can be used as (a) collectibles, such as a Pokemon from the game *Pokemon Lets Go*, or (b) custom game controllers, such as a sword-shaped controller from the *Legend of Zelda* game. (c) Examples of fabricated objects from our user study, wherein the participants integrated fabrication with existing games of their choice.

ABSTRACT

Fabricating objects from a player's gameplay, for example, collectibles of valuable game items, or custom game controllers shaped from game objects, expands ways to engage with digital games. Researchers currently create such integrated fabrication games from scratch, which is time-consuming and misses the potential of integrating fabrication with the myriad existing games. Integrating fabrication with the real-time gameplay of existing games, however, is challenging without access to the source files.

To address this challenge, we present a framework that uses on-screen visual content to integrate fabrication with existing digital games. To implement this framework, we built the *FabO* toolkit, in which (1) designers use the *FabO designer interface* to choose the gameplay moments for fabrication and tag the associated on-screen visual cues; (2) players then use *the FabO player interface* which monitors their gameplay, identifies these cues and auto-generates the fabrication files for the game objects. Results from our two user studies show that *FabO* supported in integrating fabrication with diverse games while augmenting players' experience. We discuss insights from our studies on choosing suitable on-screen visual content and gameplay moments for seamless integration of fabrication.

CCS CONCEPTS

• **Human-centered computing** → **User interface toolkits**.

KEYWORDS

Fabrication Games, Fabrication Toolkits

ACM Reference Format:
Dishita Turakhia, Harrison Mitchell Allen, Kayla DesPortes, and Stefanie Mueller. 2021. FabO: Integrating Fabrication with a Player's Gameplay in Existing Digital Games. In *Creativity and Cognition (C&C '21), June 22–23, 2021, Virtual Event, Italy.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3450741.3465239

C&C '21, June 22–23, 2021, Virtual Event, Italy

ACM ISBN 978-1-4503-8376-9/21/06.
https://doi.org/10.1145/3450741.3465239

1 INTRODUCTION

Fabricating physical objects from a player's digital gameplay expands a player's engagement with the game [3]. Creative examples

of integrating fabrication with gameplay include fabricating collectibles of valuable game items to augment the game's digital assets, custom game controllers shaped as game objects to improve the player's interactive experience, and fabricating game objects at chosen gameplay moments to learn fabrication skills.

One way to currently integrate fabrication with gameplay is by creating such games from scratch. For example, researchers have created *Destructive Games* [6] and *Terraform* [20] in which players fabricate collectibles or game controllers from their gameplay. Because creating games from scratch can be time-consuming, researchers have proposed modifying existing games with added functionalities. For example, *Blocks-To-CAD* [12] is a modified *Minecraft* game that teaches the player 3D modeling skills during the gameplay. However, modifying existing games requires access to the games' source files. Thus, while these approaches allow for tight integration of fabrication with games, they do not generalize across the large pool of existing games.

To tap the potential of myriad existing games for fabrication, we present a framework that uses on-screen content instead of accessing source files to integrate fabrication with the games. In this framework, designers choose significant gameplay moments and tag the on-screen visual cues using our system. When players play this game, our system monitors their gameplay and scans for the tagged cues. Once identified, our system extracts objects from on-screen visual content and generates fabrication files. In this way, our framework allows designers to integrate fabrication with existing games without accessing the source files that contain information on the player's gameplay, and the asset repository that contains information for generating the fabrication files.

To implement our framework with the above workflow, we developed the *FabO* toolkit, which consists of a designer interface and a player interface. Consider the example of integrating fabrication with the game *Pokemon Lets Go* in which every time players capture a Pokemon, they can fabricate a collectible of that Pokemon. To embed such a fabrication event, designers use the *FabO designer interface* to tag the on-screen text 'You have encountered a' because the same text appears on-screen every time players capture a Pokemon. When players play the game, the *FabO player interface* monitors their screen, identifies the tagged text cue, and auto-generates the fabrication files for the captured Pokemons using object extraction. Players can fabricate their collectible, for example, a Pikachu memento shown in Figure 1a.

We ran two user studies to evaluate (1) the performance and usability of the *FabO* toolkit for integrating fabrication with various existing games, and (2) the experience of fabricating objects from gameplay. In the first study, 12 participants used the *FabO designer interface* to integrate fabrication events within the games of their choice. In the second study, 12 participants played a collectible game with integrated fabrication events and used the *FabO player interface* to fabricate objects from their gameplay. Our user studies' results and the participants' feedback from the post-study interviews show that *FabO* can successfully generalize across various games. Based on our studies' findings, we discuss insights on how to choose suitable on-screen visual content and gameplay moments for integrating fabrication with existing games, and on how to integrate fabrication without hindering the gameplay.

Contributions: In summary, we contribute the following:

- a framework to augment existing games by allowing fabrication of objects from the gameplay using on-screen visual content
- a toolkit that implements our framework through a designer interface for tagging on-screen visual content and a player interface for extracting fabrication files from on-screen visual content
- insights from our two user studies, on choosing the suitable on-screen visual content and gameplay moments for successful integration of fabrication with existing games

2 RELATED WORK

Our work is related to HCI research that focuses on integrating fabrication with games, augmenting existing digital games with added functionalities, and using visual information to fabricate physical artifacts.

2.1 Integrating Fabrication with Games

Fabrication tools are pervasive today [2, 7] and allow users to fabricate objects with a fast turnaround time using laser cutters [15, 24], and with complex geometries using 3D printers [4, 14]. Researchers have recently proposed combining fabrication and games for various applications [3]. Fabrication can be integrated with games by either (1) fabricating various objects from the gameplay or by (2) having the fabrication tools be part of the gameplay. Objects that can be fabricated include physical props [19, 22], costumes [21], physical game-boards [11] and game-controllers [23, 25]. Examples to integrate fabrication tools with the gameplay include using a 3D printer to fabricate colonies onto the gameboard as the players expand their territories (*Hybrid Games* [20]). Similarly, a computer-controlled embroidery machine can be used to stitch marks over areas on the gameboard as in the strategy game *Threadsteading*[1]. Finally, laser cutters can be used to destroy physical objects as part of the gameplay so that players can use those artifacts as conversation starters during social interaction (*Destructive Games*[6]). Of these two ways of integrating fabrication with games, we focus on fabricating objects from the gameplay. However, the above examples of such games are created from scratch, which is a tedious process. To address this challenge, we explore how to integrate fabrication with existing games.

2.2 Augmenting Existing Digital Games

Digital games have existed for several decades, and new games are being created every year. Researchers have recently augmented existing digital games by modifying the games' sources code to include additional functionalities. For example, *Blocks-To-CAD* [12] is a modified *Minecraft* [8] game that allows players to learn 3D modeling during the gameplay. *BeadED Adventures* [17] is a modified *Twine* game [10] that allows players to craft beaded bracelets from their gameplay. Similarly, the *Loominary* [18] is another modified *Twine* game [10] that allows players to craft looms using a loom controller during their gameplay. Lundgren et al. [13] augment different existing digital games, such as *Zoo Tycoon* [9] with pottering. To modify these existing games researchers have modified the

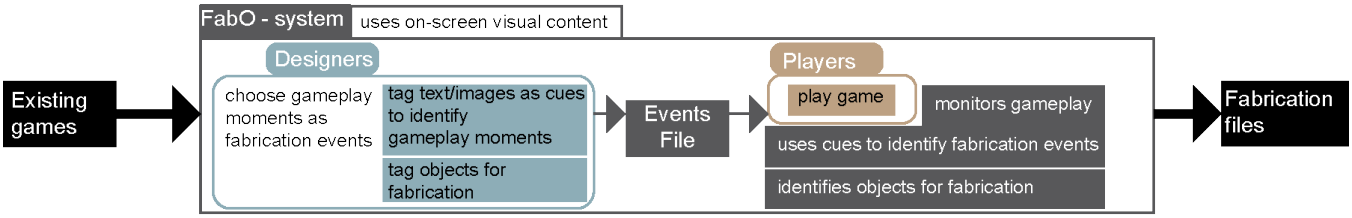


Figure 2: *FabO's* framework uses on-screen visual content to (1) allow designers to integrate fabrication with existing games and (2) auto-generate the fabrication files to allow players to fabricate objects from their gameplay.

source code. However, the source code might not always be available and thus not every game can be augmented with additional functionalities posthoc. In our work, we propose a framework that uses on-screen visual content instead of source code to augment existing games with fabrication.

2.3 Using Visual Information to Fabricate Physical Artifacts

To allow users to fabricate objects from visual content, such as images and videos, researchers have developed systems that process visual information and automatically generate fabrication files. For example, Dick et al. [5] developed a Computer-Aided Pattern-Detection (CAPD) technique that allows users to convert photos of ceramic glazing into laser cut-able patterns. In another approach, the *Mosculp* system [26] takes video of a moving human body as an input, extracts the body's motion over time from the 2D video frames, and generates a 3D sculpture that users can 3D print. In our work, to generate fabrication files, we monitor the on-screen visual content of the gameplay and process it frame by frame to find embedded fabrication events. We then identify objects to fabricate from the visual content and use object extraction to auto-generate the corresponding 2D fabrication files.

3 FABO

In this section, we first explain our framework and how it uses on-screen visual content to support integrating fabrication with existing games. We then demonstrate the implementation of this framework via the *FabO* toolkit.

3.1 FabO's Framework

We developed a framework that allows designers to use on-screen visual content to integrate fabrication with games, and the players to fabricate objects from their gameplay. As shown in Figure 2, our framework takes existing games as input and outputs fabrication files of game-objects. To achieve this, designers use our system to choose gameplay moments as fabrication events by tagging visual content as cues, such as text or images. Designers can also tag on-screen regions to extract game objects for fabrication. Using our system, the designers then export all the fabrication events in a single file. Players load this events file while playing the game. Our system then monitors their gameplay, searches for cues of the tagged events and outputs the fabrication files of the game objects. A key benefit of this framework that uses on-screen visual content rather than source code is that it can tap the potential of integrating a large pool of existing games with fabrication, where the source files may be unavailable. We accomplish this in our framework by:

Extracting Information on Player's Gameplay Without Access to the Source Code: Information on a player's gameplay, such as when they encounter significant moments in the game, can be extracted from the on-screen visual content via computer vision. Such moments are typically accompanied by visual content, such as a congratulatory message or image. For example, in the game *Pokemon Lets Go*, when players capture a Pikachu, the text message *'You have caught a Pikachu'* appears on the screen. Using computer vision to monitor and match such cues allows us to extract information that the player acquired a game object in their gameplay. Furthermore, visual content of such gameplay moments can be easily sourced from online recorded gameplay videos or by recording their own gameplay.

Generating Fabrication Files from the Gameplay Without Access to the Game's Assets: Fabrication files can be generated during the gameplay by using object extraction on the on-screen visual content via computer vision. For example, by extracting the outline of the image of Pikachu that appears on-screen in the game *Pokemon Lets Go*, we can generate its SVG file during a player's gameplay. The players can use this fabrication file to laser cut a Pikachu collectible. Alternatively, the players can mark on-screen objects for generating their fabrication files. If the object intended for fabrication is unavailable on-screen, our system allows linking a custom fabrication file that is released when players encounter that fabrication event.

3.2 FabO Toolkit

To implement the above framework of integrating fabrication with games, we developed the *FabO* toolkit that uses computer vision to extract information from the on-screen visual content. The *FabO* toolkit has two parts - *a designer interface* that designers use to tag on-screen visual cues in the game, such as text or images; and *a player interface*, that monitors the tagged cues during a player's gameplay and auto-generates fabrication files for the objects from the game. We now demonstrate the FabO toolkit's workflow with the example of the game *Pokemon Let's Go*.

Designer Interface: The first step for designers is to select an existing game, for example, we selected *Pokemon Lets Go*. The next step is to choose the gameplay moments to integrate with fabrication.

Figure 3: To integrate fabrication events, designers use *FabO's designer interface* to (a) capture the screenshot of the gameplay moment (b) tag the on-screen visual content as cues, such as text or images, (c) set event properties, and (d) choose option for object fabrication.

We choose the moments when the players capture Pokemons as the fabrication events so they can fabricate a collectible of their Pokemons.

To locate such gameplay moments, designers can either use videos of recorded gameplay from video platforms, such as Youtube, or play the game themselves. In our case, we located the Pokemon capturing moment from a gameplay video sourced from Youtube. Designers then use the 'take a new screenshot' feature to import the chosen gameplay moments into *FabO's designer interface*. For instance, Figure 3 shows the screenshot of our chosen gameplay moment of a player capturing a Pikachu imported into *FabO's designer interface*.

The next step is for designers to tag on-screen visual cues, such as text or images associated with that gameplay moment. For instance, Figure 3b shows our tagged text cue of 'You have encountered a '. We tag this cue because it allows us to create fabrication events for any Pokemon and not just a Pikachu. Alternatively, to limit our fabrication to only Pikachu, we can tag the image of Pikachu's character as a cue. Based on their preference, designers can set if the fabrication event should trigger just once or repeat every time the player encounters it. In our case, we choose to repeat the event to allow for the fabrication of Pokemons every time a player catches one (Figure 3c).

The next step for designers is to define the object for fabrication by choosing one of the three options from the designer interface: (1) specifying the on-screen area where the object appears, (2) providing a custom object's fabrication files, or (3) letting the players determine the object from the visual scene. We specify the on-screen area where Pikachu appears as it allows us to generalize for all the Pokemons that appear in that area (Figure 3d). We then save this fabrication event with its properties. The event can be modified later if needed.

After creating all the events, designers finally export the 'Fabrication Events' file which references all the fabrication events with their screenshots, the tagged events, marked regions, and choice of object's fabrication.

Player Interface: Before playing the respective game, players load the exported 'Fabrication Events' file into the player interface.

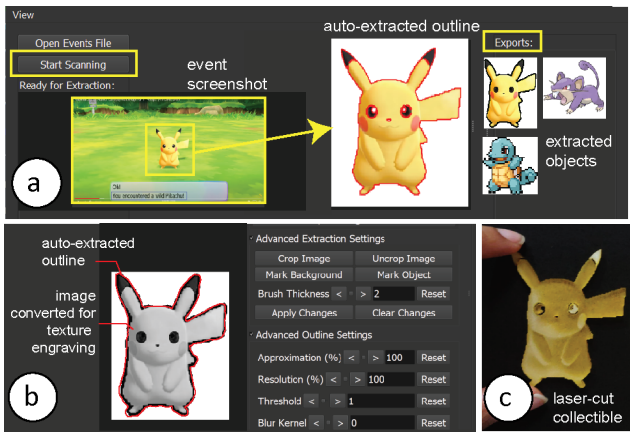


Figure 4: *FabO's player interface* monitors players' screens during the gameplay, and when they encounter a fabrication event, (a) it extracts the tagged object for fabrication from the screenshot of the event and (b) allows players to refine the extracted outline and generate fabrication files that (c) players can use with laser cutters and paper cutters to fabricate the object, for example a collectible of Pikachu from the game *Pokemon Lets Go*.

While playing, the player interface monitors a player's screen, scans for tagged cues, and identifies fabrication events. Once identified, the player interface notifies players with a prompt message. At this point, players can either continue playing or pause the game to fabricate the object from the fabrication event using the player interface (Figure 4a).

Based on the setting for the object's fabrication, the player interface either (1) automatically extracts the object's outline from the on-screen region, or (2) loads the pre-linked external fabrication file, or (3) allows players to choose an object for fabrication by marking an area on-screen.

In some instances, for example, when an on-screen object blends with the background, the auto-extracted outlines may have artifacts, such as parts of the background included, or the outline may not be smooth. In such cases, players can use the player interface (Figure 4b) to rectify the object outline by marking the respective color pixels. They can further refine the outline by adjusting parameters such as smoothness to reduce the number of points on the outline, and scale factor to increase the image's resolution, thereby reducing the artifacts due to image pixelation. After reviewing, players can export the outlines with or without image texture as SVG files for fabrication using cutting or engraving. Figure 4c, for instance, shows a Pikachu collectible that *FabO* extracted with outline and texture from an on-screen image, that we fabricated with engraved texture using a laser cutter.

FabO - Additional Features: In addition to the above workflow, the *FabO* toolkit also supports functionalities, such as sequencing the fabrication events, previewing the frequency of events, and referencing game-controller outlines within fabrication files.

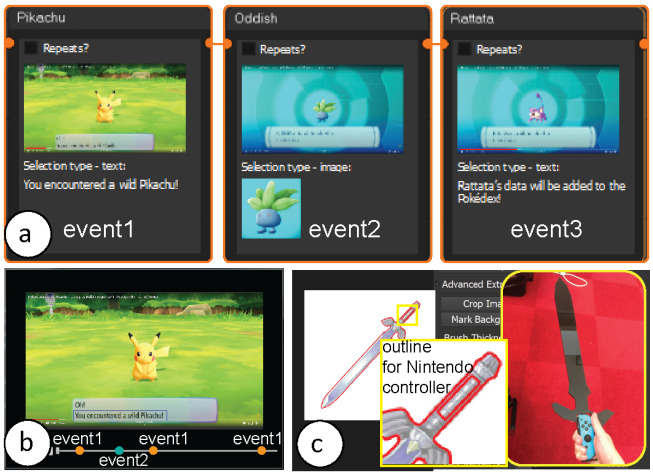


Figure 5: *FabO* allows (a) sequencing of events in a desired order to impose linearity (b) auto-scanning gameplay videos to check the frequency of the embedded events (c) inserting game-controller outlines to fit them within fabricated objects, such as a sword from the game *The Legend of Zelda*.

Designers can decide the sequence of fabrication events, such that only when a certain fabrication event has occurred, the subsequent events are unlocked. This feature allows designers to impose linearity in the fabrication of objects during the gameplay (Figure 5a). To support designers with estimating the frequency of their embedded events, the preview feature allows them to check when and how often the embedded events occur in a gameplay video by scanning the source video for tagged cues and highlighting them on the video timeline (Figure 5b). To expand the use of objects extracted from the games that use game-controllers, the player interface has a library of outlines of standard game controllers that players can overlay on their fabrication files. For example, an outline of the game controller Nintendo Switch can be combined with the outline of a sword extracted from an on-screen game object to make a personalized sword-shaped game controller for the game *The Legend of Zelda* (Figure 5c).

3.3 FabO - Implementation:

Figure 6 shows the implementation pipeline of the *FabO* toolkit which mirrors the framework described in Section 3.1. We implemented the *FabO* toolkit in Python using the *PyQt5* library as a GUI wrapper.

Designer Interface Implementation (For Extracting Information on Player's Gameplay Without Access to the Source Code): In the *FabO designer interface*, we use the Autopy library [1] to take gameplay screenshots, and display them in an editable OpenCV canvas. This editable canvas allows designers to draw bounding boxes on the loaded screenshot image and mark the text and image cues as well as select the region to monitor for cues. For text extraction from a region, we use OpenCV's thresholding function and Python-tesseract [2], an optical character recognition (OCR) tool. For processing the image cues, we use list slicing to crop the scene image to the cue region. We save the fabrication events, and the associated cues' properties in a JSON file encoded as a dictionary.

Player Interface Implementation (For Generating Fabrication Files from the Gameplay Without Access to the Game's Assets): Players can load the JSON file in the *FabO player interface* and start monitoring their gameplay. For text cue detection, we capture the screen frame by frame and then use text extraction as described above. To match the extracted text with the tagged text cue, we use the Python FuzzyWuzzy library's [3] Levenshtein Distance calculation and match accuracy to 99%. For image cue detection, we use Autopy's find_bitmap function to search a screen region pixel by pixel. Once a fabrication event is detected, we use list slicing to crop the region and then process it for object extraction using Sullivan et al.'s GrabCut algorithm [16] from OpenCV.

Additionally, the player can also tune the smoothness outline profile using settings from the UI. These settings allow adjusting the resolution (using linear interpolation), applying a bilateral filter blur kernel, applying a binary thresholding filter, and a reduction of the number of points on the outline using the approxPolyDP function. Note that all these processing steps are optional for the player. After the players fine-tune the outline, we then extract the object outline with OpenCV's contours function, and export it as an SVG file or PNG file. The player can use this file for fabricating the object using 2D paper plotters or laser-cutters.

To evaluate the performance and usability of this *FabO* toolkit, and the experience of using it to integrate fabrication with existing digital games we ran two user studies detailed in the next sections.

[1] https://www.autopy.org/documentation/api-reference/
[2] https://pypi.org/project/pytesseract/
[3] https://pypi.org/project/fuzzywuzzy/

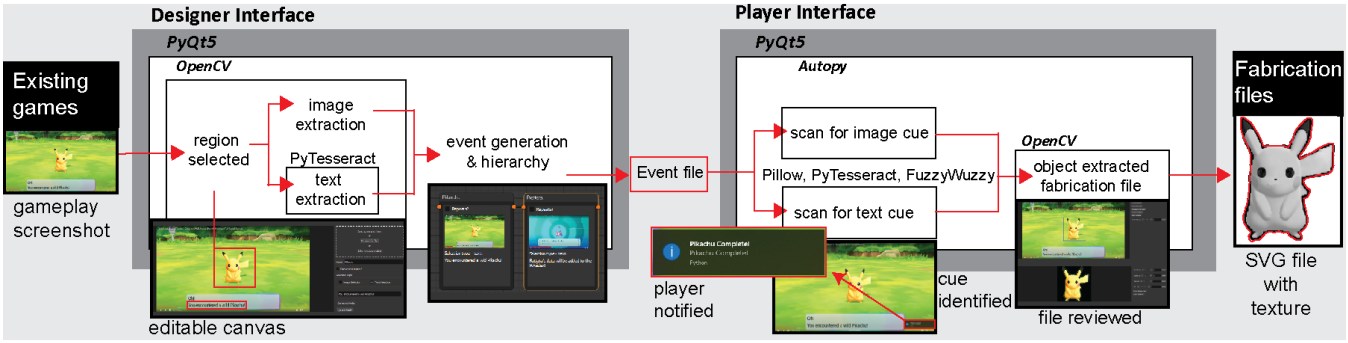


Figure 6: Implementation pipeline: The *FabO* toolkit has two components, the *designer interface* and the *player interface*, that are written in Python and use *PyQt5* as a GUI wrapper

4 USER STUDY #1: EVALUATING FABO FOR INTEGRATING FABRICATION WITH EXISTING GAMES

In the first user study, we examined *FabO's* workflow and user's experience for integrating fabrication events within various existing digital games. Insights from the study allowed us to determine how designers can choose (1) suitable visual content for *FabO's* workflow and (2) suitable gameplay moments for seamless integration of fabrication within the gameplay.

4.1 Study Design

We recruited 12 participants from our institution (6f, 5m, 1n/b) aged between 20-29 years (M=24, STD.=2.82) and with varied experience of playing digital games (10+ yrs to never playing games). We conducted the 60min study remotely over a video call (Zoom). The participants used the *FabO* toolkit on our computer via Zoom's remote control.

Before the study, we asked the participants to choose up to 3 existing digital games, gameplay moments within those games to embed fabrication events, and associated game objects for fabrication. They could source these gameplay moments either from their own gameplay or from online videos. During the study, we first demonstrated the *FabO* workflow using the game *Pokemon Lets Go*. The participants then used the *FabO designer interface* and their sourced videos to tag as many gameplay moments and associated objects for fabrication as they preferred using text and image cues. They then tested if the *FabO player interface* successfully detected their embedded fabrication events. Finally, we gathered their feedback through semi-structured interviews and a post-study feedback form.

4.2 Study Results

Altogether, the 12 participants attempted to integrate fabrication with 35 existing digital games (2-3 games per participant) across 9 genres by tagging 47 events (1-2 events per game). We tested the success of the fabrication events across three conditions: (1) were the participants able to tag on-screen visual content of their chosen gameplay moment, (2) did *FabO* identify those moments by scanning for the visual cues, and (3) did *FabO* generate a fabrication file of a game object for laser cutting. If all three conditions were met, we counted an event as a successful fabrication event.

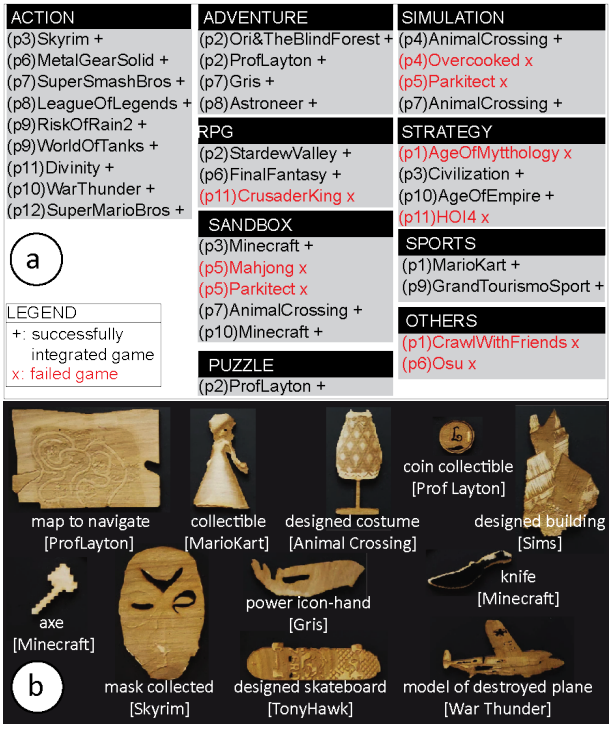


Figure 7: (a) User study participants tested 35 games (24 successful) across 9 genres. (b) Fabricated game-objects using *FabO*.

From the 35 games attempted shown in Figure 7a: (1) the participants were able to tag on-screen content for 33 games (94.29%). In 2 games, they struggled to identify a discrete moment to tag a text or image cue for integration. (2) Within the 33 tagged games, participants tagged 47 events - 15 using text cues and 32 using image cues. Of these 47 tagged events, *FabO* detected 35 events (74.47%) - 12/15 text cues (80%) and 23/32 image cues (71.19%). In

total 24 games had successfully detected events. (3) For the 35 detected events, *FabO* successfully auto-extracted the fabrication files for all game objects as marked by the participants. Thus, in total, the participants successfully integrated fabrication in 24 out of 35 games (68.57%) across all three conditions. Figure 7b shows the objects that we fabricated from the generated files. These objects ranged from commemorative trophies and collectibles to supportive gameplay tools, such as maps.

4.3 Study Insights

We studied the successful and failed examples of fabrication events from the study to gain the following insights on choosing suitable visual content and gameplay moments:

#1 Choosing Suitable Visual Content: When tagging an event, the designer has to find a text or image cue on screen that indicates that the event occurred. While most games offer such *discrete* cues through text messages or images, some games are *continuous* and do not contain such cues. An example of a game with a discrete cue is the game *Prof. Layton* [p2] (Figure 8a), in which a text message appears on-screen when players acquire a coin, thus indicating that the event occurred. However, in the 2 games for which participants failed to integrate fabrication events, i.e., *Unrailed* [p4] (Figure 8a) and *Parkitect* [p5], the gameplay was continuous with no discrete cues to indicate event occurrence, i.e., the players built a track and a park continuously, thereby making it difficult to select a discrete moment.

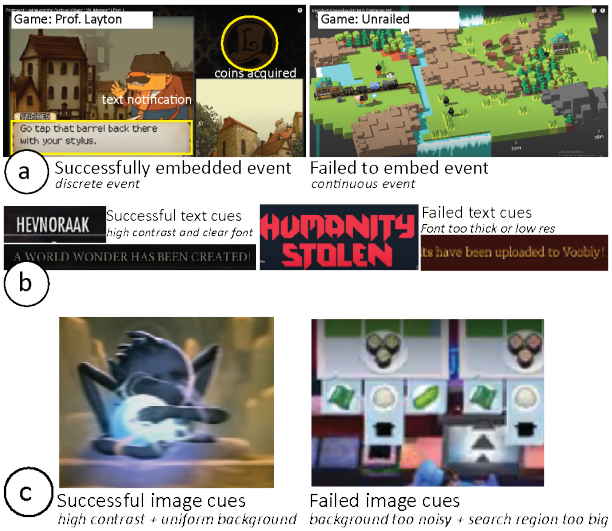


Figure 8: Examples from user study where *FabO* (a) successfully detected the text cues (that had legible text), (b) failed to detect text cues (that were pixelated) (c) onscreen text was tagged as image cue because of the font

Another important consideration in choosing the visual content is to select cues detectable using computer vision, i.e., extractable font and images with a high contrast background (Figure 8b, c). If the font was too thick, artistic or low-res (Figure 8b-bottom), the text extraction was faulty. Similarly, if the background was too noisy (Figure 8c-bottom), the image cue detection was slow and faulty. To increase the detection speed, some participants used a smaller area with less background noise for monitoring and *FabO* successfully detected the events.

Finally, to fabricate objects from visual content, it is essential to have them present on-screen at the moment selected for fabrication. When the objects were not visually present on-screen, participants linked external files with the event. However, one participant [p6] addressed this constraint by using *FabO's* sequencing feature for the game *Final Fantasy*, by using one fabrication event to trigger another event, wherein the fabrication object was on-screen. P6 wanted to fabricate the score card only after an enemy was killed. To achieve this, the participant embedded the first event where the enemy killing was identified, but did not mark an object to fabricate. However, this event unlocked the next event which was triggered when the score-card was identified for fabrication.

#2 Choosing Gameplay Moments Suitable for Seamlessly Integrating with Fabrication: When analyzing the successfully embedded fabrication events from the study, we observed that the timing of integration within the gameplay was crucial. We noted that fabrication was integrated either at the start (7/47 events, 15%), during the gameplay when there are natural pauses (31/47 events, 66%), or at the end of the gameplay (9/47 events, 19%) when the player can shift focus to fabrication.

Examples of embedded fabrication events at the start of the gameplay included moments when the player created new objects, such as a dress (*Animal Crossing* Figure 9-1), customized game-objects, such as their skateboard (*Tony Hawk* Figure 9-2), or received support objects, such as a map (*World of Tanks* Figure 9-3). Examples of embedded fabrication events during the gameplay included moments of natural pauses, either because the game paused the playing or the player paused their gameplay voluntarily. Examples of gameplay pauses included moments, such as unlocking characters (*Mario Kart* Figure 9-4) or powers (*Gris* Figure 9-6), and destroying characters or objects. Examples of player-based pauses included players updating (*Skyrim* Figure 9-7) or accessing their inventory (*Minecraft* Figure 9-8), referencing support objects, such as maps, accessing scorecards or stat cards, and socially interacting with game characters (*Ori and the Blind Forest* Figure 9-9). Because *"these are natural pauses"* [p9] when players were not concentrating on playing the game, participants chose these moments as suitable for introducing them to a fabrication activity. Examples of embedded fabrication moments at the end of the gameplay included moments when the players completed building, such as a house (*Sims* Figure 9-11), or had won the game (*Grand Tourismo Sport* Figure 9-12).

In summary, we observed that participants were mindful of the gameplay timing while integrating fabrication within it, such that it would not distract or interrupt the players while playing.

While these examples cover the various games that the participants explored, our study does not cover the full design space of existing games. Thus, more extended studies are needed to understand how to choose suitable visual content and gameplay moments for integrating fabrication with existing games while also augmenting the player's experience.

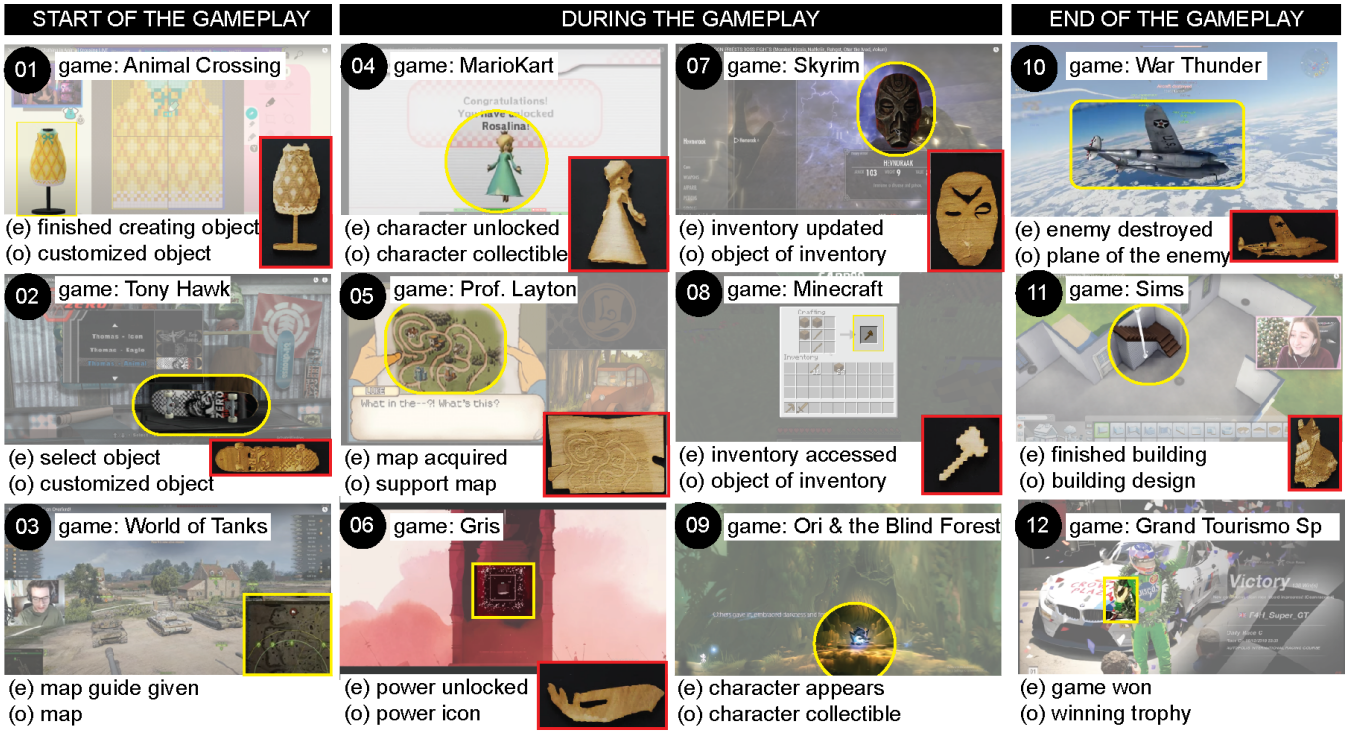


Figure 9: User study #1 examples of gameplay moments when fabrication was integrated at the start, during, or end of the gameplay. [(e): fabrication event, and (o): fabrication object.]

5 USER STUDY #2: EVALUATING THE PLAYER EXPERIENCE DURING GAMEPLAY

In the second user study, we examined player's experience of playing an existing digital game integrated with fabrication events, and then fabricating objects from their gameplay using *FabO*.

5.1 Study Design

For the study, we selected the game *Pokemon Planet* because it has a short gameplay and an open-ended story line. Using *FabO*, we embedded fabrication events that corresponded to when players received either a Pokeball or captured a Pokemon. We recruited 12 new participants from our institution (8f, 4m) aged between 17-28 years (M=22.75, STD.=4) with varied experience of playing games from few times a month to everyday. We conducted the 30min study per participant remotely over Zoom, where they played the game for 15 mins on our computer via Zoom's remote control. We did not brief the participants about the *FabO* system and simply asked them to play the game as they normally would on their own. When they encountered a fabrication event in the game, *FabO* notified them with a text prompt. At this point, they could either continue playing or pause the game to fabricate the object. For fabrication, the participants first reviewed the auto-generated fabrication files in the *FabO player interface* and then fabricated the objects, such as Pokeballs and their captured Pokemons using a remote paper-plotter via Zoom's remote control feature, and watched their objects get fabricated over the video call. We then collected feedback on their experience in a semi-structured interview and a post-study feedback form.

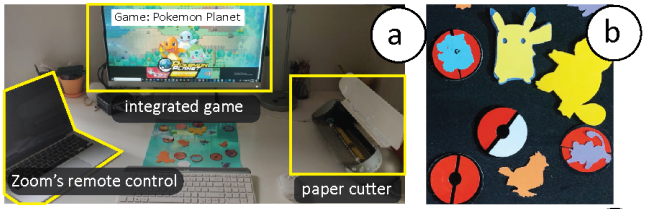
5.2 Participant Feedback

Fabrication of Objects was Meaningful: 11 out of 12 (91.6%) participants found the ability to have physical versions of digital objects from their gameplay meaningful. For example, p4 said *"There are many times during a game where I [have] thought it would be amazing to have a physical version of the equipment"* and p7 said *"I think it can be nice to build collections and to hold pride about."* However, p8 highlighted the need for closely integrating fabrication - *"The main risk of the modified game is for the fabrication event to feel out of place."* Some participants also recommended using the system for educational purposes. For example p12 said *"As an educational tool, especially for getting kids excited about fabrication, I can see it being really empowering and engaging while teaching really valuable skills in STEM."*

Choice of Objects for Fabrication: When asked if they preferred to choose which objects to fabricate and when to fabricate them, 7 out of 12 (58%) participants wanted to choose themselves. For example, p3 said *"I would love to see players given the opportunity to design and embed their own events as well - to trade in games"*. In contrast, 5 participants preferred the experience designed by someone else because it builds anticipation. For example, p10 said *"the anticipation of fabricating pokemon in real life encourages me to keep playing the game to discover new pokemon...so I can make more*

collectibles. The excitement and anticipation of playing the game and fabrication game items builds on each other." In addition, p4 said *"randomizing the fabrication events rather than having them be predictable is fun!"*

Timing of Fabrication Events: 7 out of 12 (58%) participants found the idea of fabricating objects during their gameplay enjoyable. From the other 5 participants, 3 stated that they preferred to fabricate the objects after the gameplay and not while playing the game as it halted their gameplay. For example, p2 said *"depending on the pace of gameplay, e.g., on a mission or adventure, it may feel distracting to keep having to switch out of the game to fabricate."* and p12 said *"perhaps pausing [the] game to make fab files print-ready was a bit intrusive and detracted a bit from gameplay.'*



how enjoyable was the experience of fabricating objects (while playing)?	1(not enjoyable)	2(somewhat not enjoyable)	3 (neutral)	4(somewhat enjoyable)	5 (enjoyable)
		8.3%	33.3%	50%	8.3%

did the moments of fabrication take away the fun of playing the game?	1(yes)	2(somewhat yes)	3 (neutral)	4(somewhat no)	5(no)
		25%	16.7%	58.3%	

Figure 10: (a) User study #2 setup wherein the participants remotely (via Zoom's remote control) played the game *PokemonPlanet* integrated with fabrication events, (b) fabricated objects from their gameplay using *FabO* and a remote paper-plotter. (c) Participants' feedback.

5.3 Study Insights

We thus observed that from the player's perspective, it was important that the fabrication does not hinder the gameplay and is integrated meaningfully for a definite purpose. If integrated well, our study participants' feedback shows that it may increase player's motivation, excitement, and engagement with the game without distracting their gameplay.

6 LIMITATIONS AND FUTURE WORK

Visual Cues Required: Because our framework uses on-screen visual content, we cannot extract information from moments that either (1) do not have distinct visual cues or (2) that have non-visual cues, for example sound. To address the first limitation, we can explore if machine learning techniques can be used to automatically identify significant moments and auto-label fabrication objects. For the second limitation, we can expand our system to tag and identify audio cues to include events that may not have distinct visual cues.

Trade-off in Detection Speed and Object Fidelity: Because analyzing visual content during the gameplay requires significant computation power, the speed of detection is dependent on the player's screen resolution and the processing power of their computers. While reducing the screen resolution may improve detection speed, it reduces the fabrication file's fidelity. This trade-off in performance speed and fabrication file's fidelity can be addressed by using more efficient algorithms for object detection.

2D Fabrication Only: Because we use 2D object extraction techniques for generating files, the resulting fabrication files are for 2D fabrication only. However, 3D fabrication, such as 3D printing, can also be integrated with gameplay using our framework by linking custom STL files of 3D objects to the fabrication events. For future versions of our system, we can generate 3D models from 2D visual content by (1) mapping 2D images to 3D models repositories, or (2) reconstructing the 3D geometry from 2D images through advanced graphics techniques, such as multi-view object construction.

Extending the Use of the Fabricated Objects in the Games: While incorporating the fabricated objects back into the game's mechanics is beyond the scope of our current work, it is an avenue for future work. Toolkits like *Nintendo LABO* already incorporate objects fabricated from 2D materials within games for immersive gameplay. By fabricating tangible objects and configuring them to influence the gameplay can integrate the loop of play and fabrication more tightly.

Educational and Social Maker Games: The design of our framework also allows for applications in educational and social maker games. For instance, an educator can use the sequencing feature of our toolkit to embed increasingly difficult fabrication activities for their students. Similarly, in a social setting with multiple users, every user can use the designer interface to design unique fabrication events within the game or add to each others' fabrication events. The users can then play each others' unique versions or the combined version, encounter the unique fabrication events and fabricate objects, thereby creating novel social interactions using gaming and fabrication.

7 CONCLUSION

In conclusion, we showed that fabricating objects from player's gameplay, such as collectibles, can be accomplished using our *FabO* framework, which allows designers to use on-screen content instead of source files for integration and auto-generation of fabrication files. We implemented our framework in the *FabO* toolkit and demonstrated *FabO's* workflow that uses computer vision for tagging on-screen visual cues for embedding events and extracting on-screen objects for fabrication. Through two user studies, we showed that *FabO* successfully allowed the participants to integrate fabrication with a wide variety of existing games to augment player's experience. We discussed the insights from our studies for choosing suitable on-screen visual content and gameplay moments for seamlessly integrating fabrication with the myriad existing games, thereby tapping their potential to expand players' engagement through fabrication.

ACKNOWLEDGMENTS

We thank the MIT Learning Initiative and the MIT.nano NCSoft innovations in gaming technology initiative for partial funding of this research. This work is also supported by the National Science Foundation under Grant No. 2008116.

REFERENCES

[1] Lea Albaugh, April Grow, Chenxi Liu, James McCann, Gillian Smith, and Jennifer Mankoff. 2016. Threadsteading: Playful Interaction for Textile Fabrication Devices. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. ACM, 285–288. https://doi.org/10.1145/2851581.2889466

[2] Patrick Baudisch, Stefanie Mueller, et al. 2017. Personal fabrication. *Foundations and Trends® in Human–Computer Interaction* 10, 3–4 (2017), 165–293. https://doi.org/10.1145/2909132.2934645

[3] Srinjita Bhaduri, Jesús G Ortiz Tovar, and Shaun K Kane. 2017. Fabrication games: using 3D printers to explore new interactions for tabletop games. In *Proceedings of the 2017 ACM SIGCHI Conference on Creativity and Cognition*. ACM, 51–62. https://doi.org/10.1145/3059454.3059463

[4] Xuelin Chen, Honghua Li, Chi-Wing Fu, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. 2018. 3D Fabrication with Universal Building Blocks and Pyramidal Shells. *ACM Trans. Graph.* 37, 6, Article 189 (Dec. 2018), 15 pages. https://doi.org/10.1145/3272127.3275033

[5] Nir Dick, Naama Glauber, Adi Yehezkeli, Moran Mizrahi, Shani Reches, Maiayn Ben-Yona, Anna Carmi, and Amit Zoran. 2018. Design with Minimal Intervention: Drawing with Light and Cracks. In *Proceedings of the 2018 Designing Interactive Systems Conference*. 1107–1120. https://doi.org/10.1145/3196709.3196814

[6] David Eickhoff, Stefanie Mueller, and Patrick Baudisch. 2016. Destructive games: Creating value by destroying valuable physical objects. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 3970–3974. https://doi.org/10.1145/2858036.2858113

[7] Verena Fuchsberger, Martin Murer, Manfred Tscheligi, Silvia Lindtner, Shaowen Bardzell, Jeffrey Bardzell, Andreas Reiter, and Pernille Bjorn. 2016. Fabrication & HCI: Hobbyist making, industrial production, and beyond. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. 3550–3557. https://doi.org/10.1145/2851581.2856491

[8] Minecraft Game. 2020. *Minecraft*. https://www.minecraft.net/en-us

[9] Zoo Tycoon Game. 2021. *Zoo Tycoon*. https://en.wikipedia.org/wiki/Zoo_Tycoon

[10] Twine Games. 2020. *Twine Games*. https://twinery.org/

[11] Gabriella M. Johnson and Shaun K. Kane. 2020. Game Changer: Accessible Audio and Tactile Guidance for Board and Card Games. In *Proceedings of the 17th International Web for All Conference* (Taipei, Taiwan) *(W4A '20)*. Association for Computing Machinery, New York, NY, USA, Article 9, 12 pages. https://doi.org/10.1145/3371300.3383347

[12] Ben Lafreniere and Tovi Grossman. 2018. Blocks-to-CAD: A Cross-Application Bridge from Minecraft to 3D Modeling. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*. 637–648. https://doi.org/10.1145/3242587.3242602

[13] Sus Lundgren and Staffan Björk. 2012. Neither playing nor gaming: pottering in games. In *Proceedings of the international conference on the foundations of digital games*. 113–120. https://doi.org/10.1145/2282338.2282363

[14] Stefanie Mueller, Sangha Im, Serafima Gurevich, Alexander Teibrich, Lisa Pfisterer, François Guimbretière, and Patrick Baudisch. 2014. WirePrint: 3D Printed Previews for Fast Prototyping. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology* (Honolulu, Hawaii, USA) *(UIST '14)*. Association for Computing Machinery, New York, NY, USA, 273–280. https://doi.org/10.1145/2642918.2647359

[15] Stefanie Mueller, Bastian Kruck, and Patrick Baudisch. 2013. LaserOrigami: laser-cutting 3D objects. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2585–2592. https://doi.org/10.1145/2470654.2481358

[16] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. 2004. " GrabCut" interactive foreground extraction using iterated graph cuts. *ACM transactions on graphics (TOG)* 23, 3 (2004), 309–314. https://doi.org/10.1145/1186562.1015720

[17] Anne Sullivan and Emily K Johnson. 2019. BeadED Adventures: Crafting STEM Learning. In *Proceedings of the Thirteenth International Conference on Tangible, Embedded, and Embodied Interaction*. 351–358. https://doi.org/10.1145/3294109.3300997

[18] Anne Sullivan, Joshua Allen McCoy, Sarah Hendricks, and Brittany Williams. 2018. Loominary: crafting tangible artifacts from player narrative. In *Proceedings of the Twelfth International Conference on Tangible, Embedded, and Embodied Interaction*. 443–450. https://doi.org/10.1145/3173225.3173249

[19] Joshua Tanenbaum and Karen Tanenbaum. 2015. Envisioning the Future of Wearable Play: Conceptual Models for Props and Costumes as Game Controllers.

[20] Joshua Tanenbaum, Karen Tanenbaum, and Michael Cowling. 2017. Designing Hybrid Games for Playful Fabrication: Augmentation, Accumulation and Idleness. In *Extended Abstracts Publication of the Annual Symposium on Computer-Human Interaction in Play* (Amsterdam, The Netherlands) *(CHI PLAY '17 Extended Abstracts)*. Association for Computing Machinery, New York, NY, USA, 413–419. https://doi.org/10.1145/3130859.3131334

[21] Joshua Tanenbaum, Karen Tanenbaum, Katherine Isbister, Kaho Abe, Anne Sullivan, and Luigi Anzivino. 2015. Costumes and wearables as game controllers. In *Proceedings of the Ninth International Conference on Tangible, Embedded, and Embodied Interaction*. 477–480. https://doi.org/10.1145/2677199.2683584

[22] Joshua G Tanenbaum and Karen Tanenbaum. 2015. Fabricating futures: Envisioning scenarios for home fabrication Technology. In *Creativity in the Digital Age*. Springer, 193–221.

[23] Joshua G. Tanenbaum, Amanda M. Williams, Audrey Desjardins, and Karen Tanenbaum. 2013. Democratizing Technology: Pleasure, Utility and Expressiveness in DIY and Maker Practice. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Paris, France) *(CHI '13)*. Association for Computing Machinery, New York, NY, USA, 2603–2612. https://doi.org/10.1145/2470654.2481360

[24] Tom Valkeneers, Danny Leen, Daniel Ashbrook, and Raf Ramakers. 2019. StackMold: Rapid Prototyping of Functional Multi-Material Objects with Selective Levels of Surface Details. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) *(UIST '19)*. Association for Computing Machinery, New York, NY, USA, 687–699. https://doi.org/10.1145/3332165.3347915

[25] Nicolas Villar, Kiel Gilleade, Devina Ramdunyellis, and Hans Gellersen. 2007. The VoodooIO gaming kit: a real-time adaptable gaming controller. *Computers in Entertainment (CIE)* 5, 3 (2007), 7. https://doi.org/10.1145/1316511.1316518

[26] Xiuming Zhang, Tali Dekel, Tianfan Xue, Andrew Owens, Qiurui He, Jiajun Wu, Stefanie Mueller, and William T Freeman. 2018. Mosculp: Interactive visualization of shape and time. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*. 275–285. https://doi.org/10.1145/3242587.3242592