The Verb Collective

Justin Berry
Yale University
Center for Collaborative Arts and Media
New Haven, CT United States
justin.berry@yale.edu

Bobby Berry
Yale University
Center for Collaborative Arts and Media
New Haven, CT United States
bobby.berry@yale.edu

***Abstract*—The Verb Collective is a toolkit designed to aid instructors in teaching virtual and augmented reality to artists and creatives who are exploring the medium of immersive technologies but who might have little or no background in coding. It is designed to fit seamlessly into the interface of the popular Unity development platform and, though it leverages some of the same structures of visual coding environments, it is intentionally designed without specialized interfaces. Though it has been found that visual scripting environments offer useful tools for rapid prototyping they also constrain students to the structures they employ while also teaching them a set of tools subject to change or becoming unusable if the proprietary codebase is not regularly updated. The Verb Collective is designed to be modular and lightweight and for student expertise in its functions to be directly transferable to the Unity platform as well as other similar platforms. While other popular toolkits are focused either on rapid production or on teaching code itself, the Verb Collective is focused on emergent dynamics and exploratory play with code. This model is suited to people whose interest in immersive environments and interactivity is spurred by creative pursuits. The modularity of the system is such that while it is limited in scope it is highly customizable. Designed with three tiers of users in mind, it functions as well for beginners as it does for more skilled developers. Focusing on the expressive qualities of the medium and on the the dynamic and emergent interplay of simple elements is a strategy designed to make the platform of VR and AR accessible to a broader range of people for whom novel experiences are privileged over interaction with existing industries or models.**

Keywords—virtual reality, immersive design, toolkits, rapid prototyping, education

I. Introduction

In most tools used to teach development the goal is to create students capable of coding. The tools are often based on the structures of coding itself, such as with MIT's Scratch. Indeed, this teaches strategies for coding but it does not inherently teach strategies for creating meaningful interactions or emergent dynamics that are integral to both gameplay and impactful experience. The Verb Collective is a toolkit for creating interactions and experiences in Unity inspired by a 1967-68 artwork by Richard Serra. A sculptor, Serra was interested in creating different methods for sculpture students to explore the mediums that they work with and to show how simple interactions can yield interesting results. His list of verbs was meant to show how combining different actions creates compound relationships, such as to roll to fold or, conversely, to fold to roll. Such simple combinations allowed students to quickly explore the material properties of what they were working with. An extended example would be: to push to bend to burn to twist to expand - with each verb setting the stage for the following one. For the Verb Collective we created a strategy of reducing code into very basic commands, such as 'to look'. Each verb can be triggered by another verb or can be active on start, and each verb can act as a trigger for an array of other verbs. Because each verb script has a 'core' piece of code that defines its function, it is easy for beginners to see how each individual script works. This system does not privilege the structure of code but rather the expressive qualities of a medium defined by interaction, making it an ideal toolkit for teaching VR or AR as creative mediums to artists, musicians, and game makers [1].

Fig. 1 Richard Serra's Verb List.

Many of the core problems with virtual and augmented reality as emerging mediums can be addressed through this system. One core problem is that no effective language exists to describe or fully understand these new mediums and that, as a result, they are often described or defined by the language used

for describing extant technologies. Virtual reality, for example, is often described as a gaming medium, despite its potential for purposes outside of gaming. Games often derive their power from abstraction, such as the acquisition of points or rule based interactions. Virtual reality, however, is a concrete medium, where an apple hanging from a branch is not an abstract symbol of an apple but appears to the user as an actual apple. Due to the fact that VR is so often described as a gaming medium it tends to attract creators interesting in utilizing it for the purpose of gaming. This creates a self-fulfilling promise where the people teaching VR are teaching it to gamers and their curriculums and tools are designed to facilitate this. Our goal was to create a system inspired by the exploration of concrete and material experiences as a way to open the technology to users less interested in a specific outcome, such as a game, and more interested in exploring the material properties of the worlds they create. In doing so we hope to bring in a more diverse set of makers that challenge existing notions or expectations of these mediums.

The Verb Collective grew out of an experiment with students where we tried to describe the fundamental verbs of existing mediums and to ask the question of 'what is the fundamental verb of VR?'. If the fundamental verb of film is 'to watch', the fundamental verb of music is 'to listen', and the fundamental verb of a game is 'to do' or 'to play': what is the fundamental verb of VR? We decided to operate on the hypothesis that it was 'to be'. In the process of working through that possibility we came to the conclusion that VR was a concrete medium and that it's power lies not just in what happens in the experience but in the way that its propensity for creating a sense of presence led to an increased relevance of mechanical interactions [3]. In other words, it is not only relevant in VR that you pick up an object but also how you pick up the object and how it feels in your hand. This observation led to the idea that we should break down the experience of being in VR and the process of building those experiences into smaller and more fundamental interactions that could be played with and observed for their individual properties. Each verb can be isolated and its interaction with other verbs explored to gauge its expressive potential.

In a concrete medium with the primary property of creating a sensation of being, there is apparent value in a set of tools that is not optimized for the creation of a product, but for the exploration of process. At stake is not whether you scale the mountain, but how you scale the mountain, what it feels like to do so, and that value of scaling the mountain is determined, not by the outcome itself, but by the pleasure or meaningfulness of the process, coupled by the self-generating satisfaction of being at the top of the mountain. Does the person having achieved the height of the mountain want to be there and does being there justify the use of their time? The Verb Collective is an effort to remove the anxiety of learning code while reinforcing the fact that the code is relevant not for what it is, but for what it does, allowing students and makers to focus on the core features of interactivity[1].

II. Background And Context

One excellent example of a model for diversifying the range of creators is Scratch, developed at MIT[2]. Scratch is a programming language with a visual interface that makes it easier for people with no coding experience to begin creating content of their own. It is especially appealing to young people, ages 8-16, and was designed with them in mind. While Scratch trains young students in the logic of coding and prepares them for more advanced interfaces, it is limited in its internal scope. It prioritizes the educational aspect of learning logic structures and excels at that, but it does not lend itself to a platform beyond its internal community. This is not a flaw, but it does limit the appeal of it to older students or makers that are hoping to make standalone experiences that are shareable on more distributed platforms.

A similar model that makes it easier to learn coding structures and to generate experiences is the Game Maker platform [3]. More robust in many ways than Scratch, and more easy to publish publicly, it faces the problem of being driven by a proprietary model and with a core set of functionalities explicitly oriented towards a particular outcome, that of making games. Owned by Yo Yo Games, it is a platform that came be altered or removed from the market at any time. This is true of Unity as well but, because Unity utilizes C#, many of the tools or strategies it employs are extendable to other contexts. The Verb Collective does not function outside of Unity, but extended use teaches the syntax and structure of a standalone language that can be explored in other platforms. At issue is the concern of scalability coupled with the goal of synergy with other systems and development environments.

Within Unity there are examples of well-established tools that offer simplified development and offer some accessibility to users unfamiliar with code. Two prominent examples of this are Bolt and Playmaker, both of which are visual scripting environments available on the Unity Asset Store [4]. For developers these are excellent tools to speed up production and they are also scalable so that they are capable of producing standalone projects at the professional level. Where they are less ideal is in the way that they require the learning of a unique interface that is subject to change and that does not necessarily translate to other platforms.

Often students try and learn C#, struggle and shift to Playmaker, struggle with that and switch to Bolt, and then after continuing to struggle, return to C#. The issue for them is that when they run into problems with each toolkit, they are faced with learning an entirely new system in order to overcome the problem. Exhaustion with one interface simply pushes them into the next one in the hopes that it will be easier to use. Ultimately, they return to Unity because they do not want to invest too heavily in learning something that they could not use more broadly than within those specific packages. The Verb Collective hopes to appeal to similar students whose time to invest is limited and who wish to learn the root technologies of the platform but with a structure that allows them to test ideas and gain momentum more efficiently.

Fig. 2. Examples of the different structures of verb chaining that users can create

III. Methodology

Though the techniques used in construction of the Verb Collective are not entirely novel, they were utilized based on a collection of principles aimed at making the toolkit both easy to use and engaging as a tool for experimentation in immersive design.

A. *Object-Oriented Structure*

The main functionality of the Verb Collective consists of a group of twenty scripts (the verbs) each derived from a Verb class. This Object-Oriented Programming design allows for easier comprehension by potential students/developers, as each verb is built out of the same collection of functions and structured similarly. Provided one comes to understand the Verb class itself, comprehension of any of the specific verbs comes rather quickly. In addition, each verb contains an output array of Verbs. This allows for chaining of Verbs, as upon completion the start function is called on all output Verbs, creating an easy way for developers to build sequences.

The verbs can further be separated into two subtypes: trigger verbs, which respond to user engagement, and action verbs, which direct the object to perform a specific action. The trigger verbs were built based on the common types of user interaction used for VR experiences, such as gaze input, controller touch, and button press. These triggers can then be used to initiate other verbs through their output array. Action verbs, on the other hand, are simple discrete motions that follow a structure of initialization, motion, and termination. To make action verbs easier to comprehend, they are commented in a manner that draws users to the motion section, which is the line/lines of code that make the verb function.

B. *Verb Chaining*

Due to each verb's capacity both to be activated by another verb as well as to activate other verbs upon completion there are several different verb configurations this allows users to make. We have identified four primary constructions that can be used to build interactions. Furthermore, these constructions can be used in multiple combinations to build a large multitude of complex interactions.

Linear (Fig. 2a) – The verbs are in a linear progression. Each verb plays, then activates the next verb in the sequence, until eventually reaching a final verb with no output.

Recursive (Fig. 2b) – The verbs are used to create a loop. Similar to the linear chain each verb plays, and activates the next verb, however, rather than having a final verb with no output, the final verb in the sequence outputs to the initial verb, restarting the chain.

Distributed (Fig. 2d) – The verbs output to verbs on separate objects in the Unity scene. Users can connect distinct objects by having their verbs interact/output to one another.

Dynamic (Fig. 2c) – Multiple verbs are activated at once. Since the output array of the verbs is of variable size, any single verb can activate a multitude of other verbs. This allows users to chain a single verb into multiple concurrent verbs.

C. *Tiered User Approach*

As part of the design process we highlighted three skill tiers of users that the collective is geared at working with. Setting clear progressions for user advancement and use cases provides insights on the functionalities to produce in the toolkit.

1. Tier 1
    a. Totally unfamiliar with Unity or coding
    b. Uses verbs on single objects to create single points of interaction
    c. Does not alter scripts but does adjust public variables
    d. Rarely intends to create a standalone build
2. Tier 2
    a. Some familiarity with Unity, limited code experience
    b. Creates nested interactions by accessing Verbs across objects
    c. Will read scripts and might create small modifications to the code, potentially adding their own variables or functions
    d. Wants to create standalone builds for single event or local
3. Tier 3
    a. Very familiar with Unity or Coding in General
    b. Uses few Verbs as designed but leverages triggers and other functions to speed up the development process.
    c. Write custom Verbs tailored to their specific project

d. Interested in possible public presence of a standalone build

D. Audio-Visual Content

As part of the Verb Collective we produced a set of Unity prefabs as exemplars of the types of experiences the collective is capable of building. Rather than focus on visual content these prefabs are Unity primitives with single color or gradient materials. Instead, the verb collective focuses on sound as a form of feedback. Sounds can easily be attached to any verb allowing for users to build complex sonic narratives within primitive environments. By drawing the focus away from more complex visual feedback the intent is to empower users to explore the immersive experience without concerning oneself over complex visuals.

In order to create a novel way for people to explore these interactions we built a number of environments using Unity's default polygonal GameObjects. These environments are designed to focus on encouraging or leveraging a variety of physics-based interactions such as rolling, falling, bouncing, and floating. Essentially the design is meant to take advantage of the powerful physics engine that comes with Unity in order to make dynamic and meaningful interactions readily accessible. In a traditional development context there is content or narrative to work with but in an exploratory, play based, context there is no pre-established outcome. Combining prefabricated objects and preconfigured environments with a generalized aesthetic is intended to facilitate interaction-based experiences that do not require custom models or pre-planning. Outcomes produced through this combination occupy a field of possibilities that are open ended and yet condensed enough to be recognizable and referenced by a shared discourse.

Included in the package is a sample of usable sounds that allow users to create sonic feedback without downloading additional assets. This, in combination with layered interactions, means that music generation is an easily achievable outcome. While most narrative content requires specialized models, sonic content does not, it can be used to create interactions with feedback that is satisfying on its own terms rather than through the forwarding of a pre-structured narrative. The intent of this design element is to provide opportunities for complete experiences that don't require outside resources.

IV. Conclusions

Tools that are designed for learning code and that are structured towards that purpose are useful and powerful but limited in that their primary design is not driven towards creation but rather reflection on the process of creation. Similarly, tools that are fully functional and production oriented are useful for specific pipelines but also a problem because their use is specific to an ever-changing proprietary platform. While the Verb Collective is a first attempt and possibly not the perfect toolset, it is a step towards a creation-oriented set of tools that can be used to streamline production,

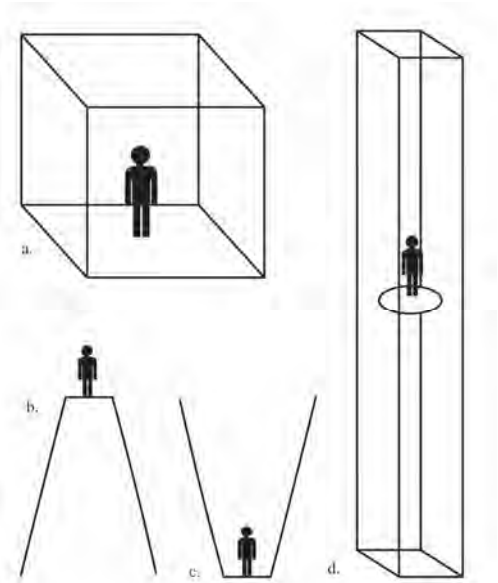

Fig. 3 An illustration of the scenes built in to the Verb Collective.

as well as education, that privileges interaction design rather than underlying mechanisms.

There is a gap in the resources available to teach people without coding experience or a development background how to generate unique interactive experiences. There is inherent value in bringing a wider audience to the development of content in emerging media because such media has not existed long enough to have a fully definable set of uses or desireable outcomes. The Verb Collective strives to close that gap and will require future iterations in order to do so in a comprehensive manner. By making the Verb Collective open source and easily available we will be able to generate enough feedback to better establish ideal paradigms for this type of resource.

References

1. Özcan, O. & Akarun, L. International Journal of Technology and Design Education (2002) 12: 161. https://doi.org/10.1023/A:1015209012458.
2. Maloney, J., Resnick, M., Rusk, N., Silverman, B., and Eastmond, E. 2010. The scratch programming language and environment. ACM Trans. Comput. Educ. 10, 4, Article 16 (November 2010), 15 pages. http://doi.acm.org/10.1145/1868358.1868363.
3. M. Overmars, "Teaching computer science through game design," in *Computer*, vol. 37, no. 4, pp. 81-83, April 2004.
4. Jere Miles, Unity 3D and Playmaker Essentials, Natick:A. K. Peters, Ltd., 2016.
5. M. V. Sanchez-Vives, M. Slater, *Nat. Rev. Neurosci.* **6**, 332 (2005).