Button Blender: Remixing Input to Improve Video Game Accessibility

Figure 1 – Over time, video game controllers have trended toward an increased number of buttons. Representative examples: Atari 2600 (1977): one button; Nintendo Entertainment System (1985): four buttons; Microsoft Xbox 360 (2005): nine buttons.

Karim Said
UMBC Information Systems
1000 Hilltop Circle
Baltimore, MD USA 21250
ksaid1@umbc.edu

Shaun K. Kane
UMBC Information Systems
1000 Hilltop Circle
Baltimore, MD USA 21250
skane@umbc.edu

CHI'13 Extended Abstracts, April 27 – May 2, 2013, Paris, France.
ACM 978-1-4503-1952-2/13/04.

Abstract
Over time, advances in video game system hardware have facilitated the evolution of video game mechanics from simple to complex. Game input devices have followed this trend, evolving from simple joysticks to multi-button, sensor-enabled controllers. Unfortunately, the complexity of modern game controllers presents significant challenges to some players, including novices and gamers with accessibility needs. Button Blender reduces such challenges by recording, remixing, and replaying game controller input, allowing novice players to play like experts.

Author Keywords
Gaming; Accessibility; Input; Remixing

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation: User Interfaces - Input Devices and Strategies

Introduction
Video games are an increasingly important part of pastime and popular culture. Video gaming offers opportunities for relaxation, competition, and camaraderie building. Moreover, the proliferation of widely varied gaming platforms and the burgeoning casual gamer market have encouraged the growth of gaming across broad segments of the population.

Figure 2 – Button Blender's workflow. Button Blender allows players to record and tweak a scripted play-through of a game and then play the script back to an emulator simultaneously with real-time controls.

Video game hardware has evolved alongside mainstream computing hardware, with increases in computing power, graphical complexity, and even new input modes. These advances have enabled game designers to create richer, more immersive experiences. To accommodate such complex games, game input devices and control schemes have become increasingly complex (Figure 1). Contemporary game controllers demand a high degree of manual dexterity and coordination, which can present a barrier to inexperienced players, as well as to players who lack manual dexterity, such as children, older adults, and gamers with motor impairments.

Towards the mitigation of these challenges, gamer communities and philanthropic organizations have emerged (e.g., The AbleGamers Foundation[1]). These communities are primarily concerned with educating gamers with disabilities and game producers, and supporting the advancement of hardware, software, and research towards game accessibility.

[1] http://www.ablegamers.com

In turn, research has focused on building frameworks for the evaluation of video game usability (e.g., [2, 4, 5, 6, 8]). Additionally, towards solving game accessibility problems, research has investigated the development of specialty hardware such as Hernandez et al's pedal system for children with cerebral palsy [3]. Researchers have also productively coopted video game devices for alternative usages, including text input with a modified joystick [7]. However, these studies all recommend extensive modification or augmentation of video game controllers or controller-like hardware, or shift their explicit focus away from games and gamers.

The present research investigates the gaps presented by this previous work. Specifically, we introduce our software framework, Button Blender, as well as a number of potential use cases, which empower players to improve video game accessibility and enhance novice play using only standard devices and widely available gaming platforms.

Button Blender

Our framework, Button Blender (Figure 2), aims to improve accessibility by recording game controller input,

remixing it to varying effect, and replaying it to a running game process. Button Blender can make novice play more like expert play, enhancing game accessibility for both novice players and players with reduced dexterity. The current prototype operates on game images running inside a game system emulator[2], which enables Button Blender to easily capture game input, remix it, and feed it to the game process. Button Blender operates on standard, unmodified games and using standard game controllers. This section describes the components of the Button Blender system and its primary features.

System Components

Button Blender uses the Snes9x[3] emulator for running game software and managing game state. Snes9x enables play of game images designed for Nintendo's Super Nintendo Entertainment System (SNES), as well as fan-made "homebrew" titles designed for that game system. We chose Snes9x due to its large existing software library and its scriptability. We developed the Button Blender prototype on the Windows 7 platform.

Button Blender currently uses a Microsoft Xbox 360 controller (Figure 1, right) for input. This controller features nine buttons, four triggers, and three directional pads. In contrast, the standard SNES system supports six face buttons, two triggers, and one directional pad. We deliberately chose a hardware controller with more buttons than the games supported so that Button

[2] While running video game emulation raises potential legal issues, emulators provide an easily instrumented environment, making it possible to capture and manipulate both game controller input and machine state. Future versions of Button Blender may use more robust virtual machine platforms, enabling Button Blender's use on a wide variety of PC games.

[3] http://www.snes9x.com

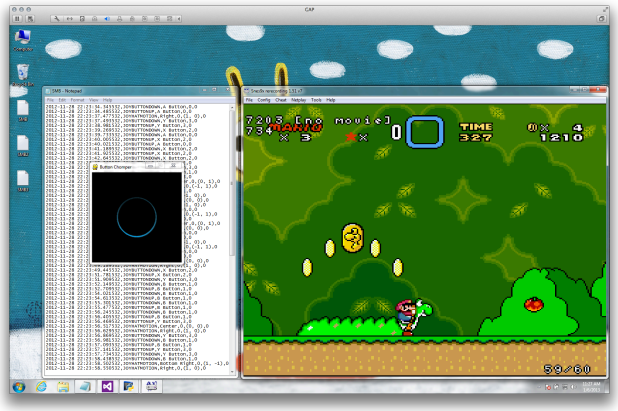

Figure 3 – Button Blender producing a play-through script during a game of Super Mario World.

Blender's additional features could be activated using the controller without affecting the underlying game controls.

While we believe that Button Blender's approach will be effective for improving the usability and accessibility for many types of games, our present exploration of the system has focused on simple, two-dimensional "platformer" games such as Nintendo's *Super Mario World* (Figure 3). This game genre features relatively simple and repetitive interactions that may be more straightforward to modify than complex 3D games. Furthermore, this genre of games was quite popular in the SNES era, and remains popular today, as seen in Team Meat's Super Meat Boy[4] and Number None, Inc.'s Braid[5].

[4] http://supermeatboy.com

[5] http://braid-game.com

Manipulating Game Input
Button Blender's primary function is to record, remix, and replay game input.

Button Blender captures game inputs via a custom Python script that runs in the background during gameplay. This script captures input event data (*e.g.,* controller ID, buttons pressed, joystick position) and stores them in a time-stamped, comma-separated log file. This file is referred to as the *play-through file.* Recording of the play-through file can be started and stopped via one of the unused controller buttons.

These play-through files can be loaded back into Button Blender and used to manipulate data in a subsequent game. Button Blender uses the Microsoft Windows API to capture, override, and emulate controller input. Thus, Button Blender can capture a button press made by the player, replace it with an alternative button press from the play-through script, and pass the replaced button event to the emulator. The following section illustrates how this core functionality can be used to provide gamers with various types of play assistance.

Usage Scenarios

The *Record-Remix-Replay* architecture provided by Button Blender can be applied in various ways to enhance video game usability and accessibility. This section describes several "scripts" that are supported by the Button Blender prototype.

Sticky Buttons
People with motor impairments often have difficulty pressing multiple keys simultaneously. "Sticky keys" are a popular keyboard interface feature [1] that allow a user to press two keys in sequence, rather than hold one key while pressing another. Button Blender can automatically add sticky-key-like functionality to existing games by "locking" a button. For example, holding down the "walk forward" key for several seconds will lock this key, causing the player's character to continue to walk forward even if the button is not being held down, and until the button is pressed again. This feature allows a player who has difficulty pressing multiple buttons simultaneously to free her hand and more easily press other buttons.

Make Any Game a One-Button Game
A *one-button* game is a game that requires the user to only press one button to play the entire game. For example, in *Halfbrick Studios' Jetpack Joyride*[6] the player's character automatically walks forward at a fixed pace, while the player uses a single button to make the character jump over obstacles. These one-button games are often considered to be accessible, as they do not require complex button combinations to be pressed. However, relatively few one-button games exist. Button Blender can turn a traditional game into a one-button game by recording an expert's play-through file, filtering out all instances of a certain button press (e.g., pressing the "jump" button), and combining replay of the script with the novice's input. In this way, the game proceeds as if the expert player were moving the character, but with jump events controlled by the novice player.

Tag Team Play and Expert Assistance
Sometimes, a game player who experiences difficulty with a certain segment of a game may seek help from a

[6] http://halfbrick.com/our-games/jetpack-joyride

more experienced player, such as a friend or family member. Button Blender's ability to combine and remix input enables this expert friend or family member to assist either synchronously or asynchronously. In *tag team play*, Button Blender can allow two players to take turns controlling a single character. In this mode, pressing a specified button switches input from one game controller to another. Tag team play can be used to enable one player to help a player through a difficult section, or to make a single-player game more collaborative.

As the expert player may not always be available to assist the novice, *expert assistance* enables the user to replay a previously recorded play-through segment. In expert assistance mode, the expert player records her input at a certain difficult section of the game. Later, the novice player reaches the same section of the game, and activates the expert's play-through. Button Blender replays the expert player's input to enable the novice user to pass through the difficult game area.

Limitations and Future Work

The Button Blender project is attempting to solve a difficult problem: that of making existing video games easier and more accessible without altering the underlying game code or controllers. As such, our implementation faces several challenges and limitations that will shape our future work.

Input and Gameplay Lag

Button Blender uses separate processes to capture input and manage the game input. This can result in synchronization errors, when the game emulator falls behind the input capture, or vice versa. Furthermore, running the emulator and input capture script result in a high system load, which can result in slow or jittery gameplay. These issues may be addressed to some degree by increasing processing power of the system far beyond that required by the game emulator, or by reducing the performance hit of the input capture script. Embedding Button Blender's functionality directly into the game emulator, rather than relying on separate scripts and inter-process communication, may enhance overall system performance.

Matching Game States

Several of Button Blender's features, such as expert assistance, require matching the player's current game state to the play state of the expert's play-through. Implementing this feature robustly requires the ability to identify when two game players are at the same approximate location in the game world. Currently, we assume that the novice player can identify their approximate location and match their current state to the appropriate expert recording. However, a more robust state-matching method would significantly extend Button Blender's capabilities.

We are exploring several techniques to identify and match game states between the live game and recorded play-through data. Button Blender's emulated architecture allows us to save and analyze the memory of the emulated game system at any point. However, analyzing large chunks of system memory may be infeasible in real time, and the sections of memory that correspond to game location may vary across games, or even across sections of a single game. We are also exploring computer vision-based matching techniques, although game location may not always be identifiable from the game image alone. Solving the state-matching problem will likely require a hybrid approach. We intend

to explore this issue by collecting data from additional games, and by collecting input data from real players.

Player Enjoyment and Game Difficulty

The core objective of any video game is to be both enjoyable and challenging. While the potential use cases of Button Blender improve accessibility and enhance game play for the novice, these changes may go too far in reducing the challenge of the game, and thereby reduce the player's enjoyment. One potential solution is to enable varying levels of assistance. Furthermore, the appropriate level of assistance might vary between players, and may change as a player becomes more familiar with a game. Testing Button Blender with game players who might benefit from using it will help us better understand the tradeoffs between difficulty, accessibility, and enjoyment.

Conclusion

Video gamers who encounter difficulties playing through a game may rely on a more experienced friend or family member for help, handing over the game controller for assistance through the difficult game area. Our prototype, Button Blender, provides a similar form of assistance to game players, combining their input with *recorded gameplay input* from an expert user, or from a pre-recorded script. This approach of recording, remixing, and replaying game input can be used to support video game accessibility and enhance novice gameplay for a variety of games without changing the underlying code, converting previously challenging mainstream games into accessible games.

References

[1] Brown, C. 1992, Assistive technology computers and people with disabilities. Communications of the A.C.M., 35 (5), 36

[2] Clanton, C. 1998. An Interpreted Demonstration of Computer Game Design. *Proc CHI '98*, 7–8.

[3] Hernandez, H.A., Graham, T.C.N., Fehlings, D., Switzer, L., Ye, Z., Bellay, Q., Hamza, M.A., Savery, C. and Stach, T. 2012. Design of an Exergaming Station for Children with Cerebral Palsy. *Proc CHI '12*, 2619–2628.

[4] Pinelle, D. 2008. Heuristic Evaluation for Games. *Proc CHI '08*, 1453–1462.

[5] Pinelle, D., Wong, N. and Stach, T. 2009. Usability Heuristics for Networked Multiplayer Games. *Proc GROUP '09*, 169–178.

[6] Pinelle, D., Wong, N., Stach, T. and Gutwin, C. 2008. Using Genres to Customize Usability Evaluations of Video Games. *Proc. Future Play '08*, 129–136.

[7] Wobbrock, J.O., Myers, B.A. and Aung, H.H. 2004. Writing with a Joystick. *Proc Graphics Interface '04*, 1–8.

[8] Yuan, B., Folmer, E. and Harris, F.C. 2011. Game Accessibility. *Universal Access in the Information Society*. 10, 81–100.