A Video Game-Based Framework for Analyzing Human-Robot Interaction: Characterizing Interface Design in Real-Time Interactive Multimedia Applications

Justin Richer and Jill L. Drury
The MITRE Corporation
202 Burlington Road
Bedford, MA 01730-1420 USA
+1-781-271-2000
{jricher, jldrury}@mitre.org

ABSTRACT
There is growing interest in mining the world of video games to find inspiration for human-robot interaction (HRI) design. This paper segments video game interaction into domain-independent components which together form a framework that can be used to characterize real-time interactive multimedia applications in general and HRI in particular. We provide examples of using the components in both the video game and the Unmanned Aerial Vehicle (UAV) domains (treating UAVs as airborne robots). Beyond characterization, the framework can be used to inspire new HRI designs and compare different designs; we provide an example comparison of two UAV ground station applications.

Categories and Subject Descriptors
H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – *auditory feedback, graphical user interfaces, haptic I/O, input devices and strategies, interaction styles, screen design, voice I/O.*

General Terms
Documentation, Design, Human Factors.

Keywords
Human-robot interaction, HRI, unmanned aerial vehicles, UAVs, interaction design, evaluation.

1. INTRODUCTION
Until recently, most robots were used in the laboratory or in real-world situations by the robots' developers, so their interfaces could be complex and require significant training. Robots are now being used more frequently by non-roboticists, however. As robots such as the Roomba™ vacuum cleaner become popular and the military relies more heavily on Unmanned Aerial Vehicles (for example), many different types of people are using robots without direct help from their developers. Accordingly, there is a need to improve human-robot interaction (HRI) so that the intended end-users can more easily employ robots.

Our goal is to improve users' interactions with airborne robots, known as Unmanned Aerial Vehicles (UAVs), to answer a critical military need. Many more "mishaps" resulting in damage or loss of aircraft have occurred per flight hour for UAVs than for inhabited aircraft, and more than half of the mishaps have been attributed to poor human factors (Tvaryanas et al., 2005).

As part of our approach to understanding possible improvement to UAV interfaces, we have been analyzing interfaces in the mature field of video gaming. Similar to UAV operations, many video games require players to understand where important objects are in a 3D environment and undertake fast-paced activities that require efficient interaction. Jørgensen (2004) notes that the fields of human-computer interaction (which is closely related to HRI) and video games should inform each other to a greater extent.

Successful video games are able to provide players with needed information and control capabilities in an engaging and enjoyable fashion. Video games are effectively streamlined input-output systems, and to a player, a video game is little more than its interface (Pausch et al., 1994). Unlike most computer applications in which the interface serves as a means of interacting with some underlying functionality, the sole purpose of a video game's interface is for the player to interact with it. This is further illustrated by the fact that video games with frustrating or cumbersome interfaces seldom succeed in the marketplace. There is a strong impetus for game interfaces to be well-designed, and therefore there is strong motivation for researchers to mine the world of video games for new ideas in interface design, especially in the highly dynamic, multimedia world of robotic interfaces.

Video games have been used by researchers for several purposes. For example, one video game [*Doom*] was used as inspiration for process management interaction (Chao, 2001). A few researchers have used video games as inspiration for human-robot interfaces. Maxwell et al. (2004) designed their robot's interface with the "First-Person Shooter" (FPS; a combat-oriented game presented from the viewpoint of the main character) genre in mind. Jones and Snyder (2001) used an architecture based on a real-time strategy game interface paradigm. Tejada et al. (2003) used a variant of the [*Unreal Tournament*] engine to develop an interface for a team of urban search and rescue (USAR) robots; Lewis et al. (2003) used the same engine to develop a simulation of National Institute of Standards and Technology (NIST) Reference Test Facility for Autonomous Mobile Robots. While these efforts have

Human Robot Interaction'06, March 2-3, 2006, Salt Lake City, Utah USA.

capitalized on video game experience, they have not included a broad look at how different video game interface approaches might relate to and inform HRI.

We have developed a framework of interface components that is based on a survey of video game designs. By breaking down video game interfaces into their components and generalizing them into a framework, researchers can use that framework to talk about interfaces for UAVs and other robots and identify promising interaction design approaches.

The remainder of this paper is divided into five sections. The next section provides an overview of the framework. Section 3 describes the framework. Section 4 describes the use of the framework to compare two different interfaces developed for the same application: the operator control interface of a small UAV. Discussion and conclusion sections complete the paper.

2. OVERVIEW

Our approach has been to examine the user interaction afforded by a wide range of video games and to describe this interaction in terms of the constituent components of the various interfaces. We then abstracted the description of the video game interaction into categories that could potentially apply to HRI. The primary purpose for developing this framework is to have a structured, reproducible means of characterizing proposed HRI designs. We anticipate the framework will be particularly useful for understand the differences in interaction approaches of different designs. The secondary purpose of the survey of video game user interaction that underlies this framework is to expose to the HRI community the wide range of interaction types being used successfully in the video game community.

Others have developed taxonomies of video game user interfaces such as Ye (2004) and Wolf (2001). These taxonomies discuss video games at the genre level, such as "Shooter" (a combative game in which a player shoots at enemies) and "Vehicular Simulator" (a generally non-combative game focused around the piloting of a virtual vehicle). While genre categories such as Shooter are useful for classifying different types of user experiences, they do not necessarily identify unique user interface approaches. For example, a First Person Shooter game may make use of an internal object-centered camera (defined and described below), as does a Vehicular Simulation game. Also, a different Shooter game in the same genre as this FPS game could use an external camera (also defined below) instead of an internal object-centered camera: thus the games' user interaction designs cannot be used as discriminators when employing a genre-based categorization. Our framework differs from a genre-based approach because it focuses specifically on the user interface and examines these user interaction components at a fine-grained level. Further, our framework does not assume a hierarchical structure nor does it assume that its components are mutually exclusive (hence it is a framework and not a taxonomy).

We developed the framework by starting with a classic division of interface components: input versus output. Quite simply: is a part of the interface chiefly employed by the user to enter information into the system, or to obtain information from the system? Another basic division of concepts related to user interfaces consists of the input or output *device* versus the input or output *method* used to interact with that device. Finally, we developed a fifth major category to include a further characterization of input methods by noting the object (or "target") of the input action and the complexity of the input method.

A high-level view of the framework can be seen in Table 1 (numbers indicate applicable section numbers in this paper). While we define the components in the following section in terms of the video game domain, we provide numerous examples regarding how the interface components may be manifested in (usually hypothetical) UAV interfaces. An italicized name in square brackets indicates a video game title that exemplifies the concept being discussed.

3. THE VIDEO GAME-BASED FRAMEWORK (VGBF) FOR HRI

3.1 Output Devices

Video screens (denoted 3.1.1 in Table 1), speakers (3.1.2), and haptic devices (3.1.3) comprise the means through which games communicate with players. Due to their reliance on direct contact with the player for transmitting information, haptic devices are most often housed within the game controller. "Controller" is used throughout this paper as a generic term for a physical device that is composed of one or more input devices. As a first step, interfaces can be characterized by the numbers and types of output devices available to the user.

Table 1. Overview of the Video Game-Based Framework for Characterizing HRI
(Numbers are keyed to sections in this paper)

3.1 Output devices	3.2 Output methods	3.3 Input devices	3.4 Input methods	3.5 Input classifications
3.1.1 Video screens 3.1.2 Speakers 3.1.3 Haptic devices	3.2.1 Primary animated graphical visual output • 3.2.1.1. Camera perspective • 3.2.1.2. Camera movement 3.2.2 Additional visual output • 3.2.2.1 Type • 3.2.2.2 Location • 3.2.2.3 Temporality 3.2.3 Non-visual output • 3.2.3.1 Audio • 3.2.3.2 Haptic	3.3.1 Buttons 3.3.2 Joysticks 3.3.3 Pointing devices 3.3.4 Multimodal 3.3.5 Specialized controller	3.4.1 Command 3.4.2 Natural language 3.4.3. Cursor 3.4.4 Camera control	3.5.1 Target 3.5.2 Complexity • 3.5.2.1 Simple • 3.5.2.2 Contextual • 3.5.2.3 Combina-tional • 3.5.2.4 Sequential

3.2 Output Methods

3.2.1 Primary Animated Graphical Visual Output

In nearly every video game, the primary animated graphical output, colloquially known as "video," is highly important.

To view the game world, there must be a "camera": a point and direction from which the view of the world is projected (as opposed to a physical video-input device). This point is also known as the "view reference point" (VRP) (Foley et al., 1990). Video output is classified by the camera and its relation to the game environment. Further, the position and orientation of this camera within the game world defines the type of animated graphical output in a given game. Animated graphical output is so central to most game experience that several game genres are named after the nature of the camera, such as FPS and Third-Person Action/Adventure.

The camera's view into the game world is defined by a set of vectors which describe the origin point of the view and the direction in which the view is facing. The VRP is combined with a "view plane normal" (VPN) and "view-up vector" (VUP), both of which project out from the VRP orthogonal to each other and serve to define the orientation of the camera in space (Foley et al., 1990). Specifically, the VPN is directed toward the point the camera is looking at, while the VUP is directed vertically upward in the camera's field of view. Coupled with information about the type of projection being done, these camera vectors, illustrated in Figure 1, define what is needed to build a view into the game world. The classifications of camera systems within video games largely relate to the placement (which yields perspective), movement, and control of these vectors, as described below.

3.2.1.1 Camera Perspective

Camera perspective denotes the viewpoint of the virtual camera lens through which a user sees the world. There are four main approaches to camera perspectives, as defined in Table 2.

3.2.1.2 Camera Movement

The ways in which a virtual camera's position and orientation are allowed to change, whether in relation to input from a player or a game event, are also defining characteristics. The four types of camera movement are described in Table 3.

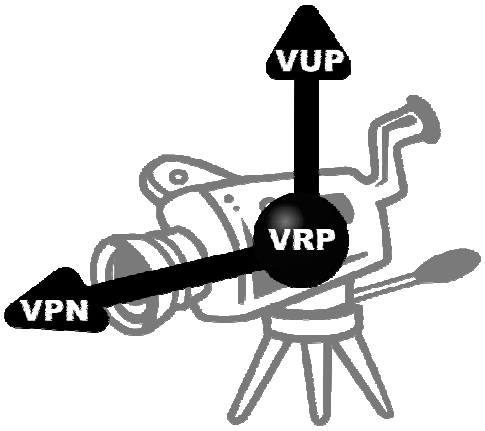

Figure 1. Important Vectors for Defining Camera Views

3.2.2 Additional Visual Output

Status and meta-information about the game and game environment constitute other types of visual output. More specifically, these types of additional visual information may consist of resources that are consumed or degraded (e.g., ammunition in a shooter game, battery life for a UAV), environmental information (e.g., overhead maps), and status (e.g., the name of the current level of game play, the name of the next waypoint for a UAV). The information may be displayed continuously, when a change takes place, or when requested. There are five locations where the additional visual information might be displayed: in a "heads up" display, attached to an object, in a subinterface, in a secondary display, or integrated into the environment somehow (called narrative integration).

Heads up displays overlay information on top of the main video view, such as when a score is displayed in the top right corner of the video screen [*Super Mario World*]. In object attachment, information is located visually close to the object to which it pertains; for example, the relative strength of two units is displayed directly above the avatars for these units [*Advance Wars*]. Subinterfaces are visual display spaces separate and distinct from the primary display, such as an inventory screen that shows a list of all items a player's character is currently carrying [*Deus Ex*]. When it is displayed, a subinterface may occlude portions of the primary animated graphical visual output display. Secondary displays are a physically distinct visual display. In some games, individual player statistics are available simultaneously on a separate screen, leaving the main screen available for the primary display [*Final Fantasy: Crystal Chronicles*]. Narrative integration of information displays meta-information by altering the display of objects within the game to reflect a current state. An example of narrative integration is a character that is animated as breathing more heavily when it reaches the limits of its stamina [*Eternal Darkness: Sanity's Requiem*].

3.2.3 Non-Visual Output

Non-visual output consists of audio output (3.2.3.1 in Table 1) and haptic feedback (3.2.3.2).

Audio output may take the form of sound effect, music, or voice. We consider a sound effect to be any audio sound that is neither music nor voice. An example of a sound effect in the video world is a chime that is played every time a player picks up an object [*Katamari Damacy*]; a UAV operator might hear an alarm if the aircraft altitude has fallen too low for normal operations. We define music to be generally continuous recognizable patterns of sound, and different background music is played for each distinct level in a game in the case of [*Super Mario Sunshine*] (for example). A use of voice in a video game might occur when a player's character has a conversation with another, allowing the player to find out something new about the game plot [*Half Life 2*]. For UAVs, a system alarm might warn of critically low velocity by using a recorded voice clip. Further information may be conveyed via the direction the sound seems to be originating from.

Haptic feedback conveys information to the player through touch or pressure. Haptic feedback can be subdivided into active resistance, passive resistance, and vibration. Active resistance occurs when the game player or UAV operator attempts to input a

Table 2. Camera Perspectives (Paragraph 3.2.1.1)

Type	Definition	Video Game Example	UAV Example
Internal attachment to a primary object	The virtual camera is directly attached to a primary object to the point of being embedded within it.	The virtual camera is implanted in the head of the virtual character, looking forward out of their eyes, and the VUP pointing upward from the character's head [*Unreal Tournament*].	UAV sensors are usually literally housed within the UAV, so when operators view sensor information from the UAV, they often view it from the perspective of the UAV.
External attachment to a primary object	The virtual camera is tied to a primary object but is placed at some distance from that object.	The virtual camera follows the primary character so that it is in full view at all times [*Pimkin*].	The operator could view a representation of the UAV from behind, as if a virtual camera were mounted in a chase plane.
Not attached to a primary object	The virtual camera is tied to something in the environment rather than a primary object, or is not attached to anything.	A player sees the environment via a virtual camera that is a corner of a room [*Alone in the Dark*].	An interface could show multiple UAVs from a high point in the sky over the intended area of operations.
Multiple perspectives	Perspectives may be switched between various types for different situations or multiple virtual camera viewpoints may be present simultaneously.	A game may allow a player to switch between seeing from the eyes of a character (internal attachment to a primary object) to looking at the character and his/her immediate environment from slightly behind the character (external attachment to a primary object) [*URU: Ages Beyond Myst*].	Most current UAV interfaces allow the operator to switch between a top-down view (camera not attached to any object) and streaming video view (internal attachment to a primary object).

Table 3. Camera Movement (Paragraph 3.2.1.2)

Type	Definition	Video Game Example	UAV Example
Fixed	A virtual camera that has been affixed to a point in the environment or to a particular object.	A virtual camera that been internally attached to a primary object is, by definition, fixed to that object and thus camera movement depends upon the movement of the primary object [*Halo 2*].	The streaming video view in a UAV interface is normally the result of a physical video camera that has been internally attached to the UAV (especially for non-gimballed cameras); thus the video camera movement is fixed relative to the UAV.
Free	A virtual camera that can be moved throughout the environment with few restrictions.	A virtual camera that is from the viewpoint of an all-seeing being may be moved in virtually any direction or orientation [*Black & White*].	In a hypothetical UAV interface, an operator may move a virtual camera such that the aircraft is viewed from any angle or that any point on the ground (including map-based synthetic terrain) is in view.
Mixed mode	The situation in which camera movement changes do not fall into a single category.	A free camera may be used to navigate to part of the environment and then becomes fixed to a selected object [*Sid Meier's Civilization II*].	A hypothetical UAV interface may include a camera that is fixed to a primary object that streams information into a video window but also have free camera view of the aircraft and its synthetic environment.
Rail	A virtual camera that moves freely along a fixed course, as if on a set of rails.	A virtual camera's orientation (VPN) is allowed to change freely but its position is constrained by moving along a virtual rail [*Ico*].	In a hypothetical UAV interface, a view of the area around the aircraft could be constrained such that the camera moves freely along a circle of fixed circumference around the aircraft.

command using input hardware yet the hardware physically pushes back against the human's command. Passive resistance consists of the physical characteristics of an input controller that convey to the player the current state of the input devices, such as when buttons embedded in a UAV controller remain depressed after activation so they convey state information via touch. Vibration consists of shaking a physical device in response to a command, and might be employed in a UAV joystick when the operator attempts to make a maneuver outside the aircraft's flight envelope.

3.3 Input Devices

Input devices detect interactions from the player and input these interactions into the game world. There are five types of input devices: buttons, joysticks, pointing devices, multimodal devices, and specialized controllers.

3.3.1 Buttons

Buttons are devices which the player may push or pull to invoke some command or state. Surface buttons are mounted on the surface of a controlling device designed to be activated by

pushing with the fingers or thumbs. Keyboards are a large array of surface buttons, usually arranged closely together in a manner to facilitate textual input. Triggers are variants of the surface button that are designed to be pulled by the fingers (as opposed to pushed by the thumbs or fingers) and are generally located on the backside or underside of a controller.

3.3.2 Joysticks

Joysticks are direction-relative input devices allowing one or more dimensions of input. They can be characterized by their degrees of freedom and whether they are digital or analog. Degrees of freedom are the number of directions of movement and orientation (dimensions) that a single joystick device is capable of detecting. A digital joystick can detect a discrete number of directions but not the magnitude of any input. An analog joystick can detect both the magnitude and direction of an input along a given axis.

3.3.3 Pointing Devices

Pointing devices are means of positioning a cursor within an interface. They are characterized by whether they indicate location by either by absolute or relative positions of the device. An absolute position is when the location of the pointing device is analogous to the position of the cursor being controlled. A relative position is when the movement of the input device corresponds to movement of a cursor.

3.3.4 Multimodal Devices

By "multimodal," we mean input systems that do not require direct physical contact from the player: they receive input via audio or visual means. Audio input systems translate an audio stream received via microphone to a game control [*Nintendogs*], and video input systems use a visual sensor to input a video stream to the game software. A camera pointed at the player may detect the player's movements, and the game responds accordingly [*Eye Toy: Play*].

3.3.5 Specialized Controller

A specialized controller is an input device that is unique to a particular game, such as a pair of drums that the player drums on in rhythm with a musical score [*Donkey Konga*].

3.4 Input Methods

Input methods are the means for a game to use the signals produced by an input device. There are four main types: commands, natural language, cursor, and camera control.

Commands can be thought of as a direct connection between an input and an output response; picture a button push causing a character to jump [*Super Mario Sunshine*]. Natural language consists of input that uses spoken or written human language; for example when a player types "pick up book" and the character then picks up a book. Many UAV interfaces currently rely on the operator to type new airspeeds or altitudes. The cursor input method is a positional pointer located over a piece of the game world, and is used, for example, in [*WarioWare Touched!*] to draw a symbol on a touch screen to access a particular command. Camera control is the amount of influence that a player has over the motion of a camera within the game environment.

3.5 Input Classifications

This category describes how the various input methods interact with objects within the game world, and can be characterized by the input's target and complexity level. The target (3.5.1 in Table 1) consists of what an input mechanism affects, and whether the player can control it directly. An example of direct control of a primary object is when the player pushes a button to command a character to jump. That jumping action could activate a switch in the environment [*Super Mario Sunshine*], an example of indirect control of the switch.

Complexity (3.5.2 in Table 1) can be described in four different ways: simple, contextual, combinational, or sequential. Simple inputs are when the player performs a basic action, such as a button press, the game performs the appropriate response. A contextual command is a command that has different effects in different situations. For example, a single button command could be used to both draw a sheathed sword and attack with it once drawn [*The Legend of Zelda: Ocarina of Time*]. In the UAV world, a button for "take off" could change to another function such as "return home" once the aircraft is airborne. In a combinational command, several simple commands activated simultaneously could serve as a new command. Pressing one button causes the character to punch, while pressing another button causes a character to block; yet when pressed at the same time, these two buttons together cause the character to throw [*Dead or Alive 3*]. Finally, several simple commands entered in sequence within a certain time period of each other are considered a sequential command. An example of a sequential command can be found in [*The Legend of Zelda: The Wind Waker*]: a dodge command immediately followed by an attack command produces an evasive strike that neither command separately could produce.

4. EXAMPLE OF USE: COMPARING TWO UAV INTERFACES

As an example of how the VGBF can be used, we characterize two alternative interfaces that are intended to be used with the same UAV: a 4' wingspan, foam, flying wing equipped with a fixed video camera, GPS, and a Kestrel autopilot, purchased from Procerus UAV, Inc. With the aircraft, we also purchased Procerus' Virtual Cockpit (VC) ground station. The VC can either display an artificial horizon and a complete set of scrollable flight parameters (see a screenshot of the left hand side of the screen in Figure 2) or operator-entered waypoints on top of an image of a park in Provo, Utah (which cannot be swapped out for the customer's operational area without a programming change).

In conjunction with Brigham Young University's Human-Centered Machine Intelligence Lab run by Prof. Michael Goodrich, we developed an interface for the Procerus UAV that is intended to be used by an operator working with live video downlink information from an aircraft. Our interface, called the Augmented Virtuality Interface (AVI), uses imported terrain data and geo-references the streaming video subwindow so that its live data matches the location of the imported terrain data. AVI uses a "chase plane" view of a UAV avatar so the operator can see the attitude of the aircraft as it banks, dives, or climbs in real time. AVI can be seen in Figure 3.

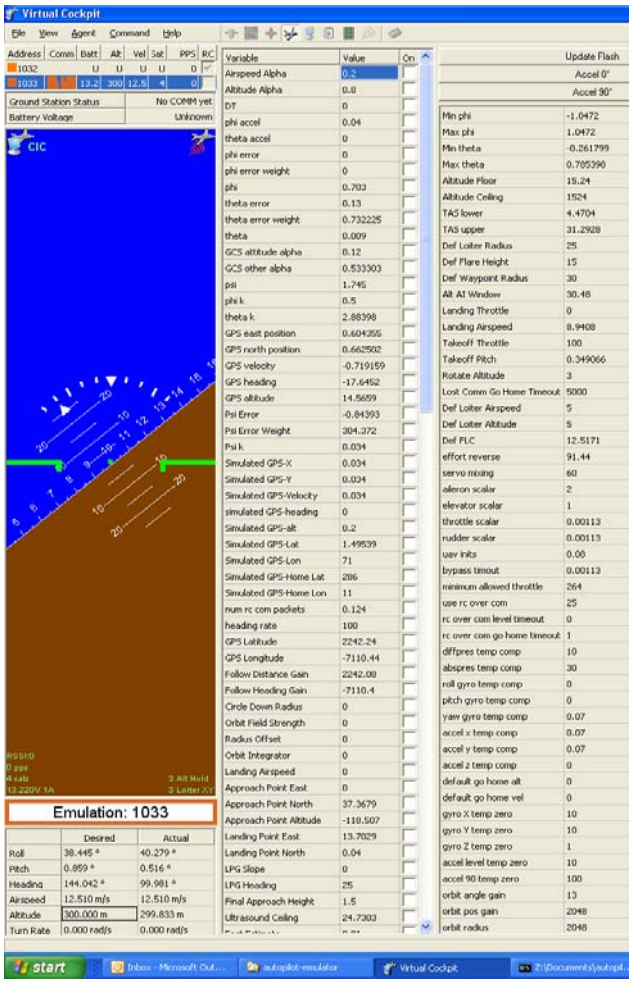

Figure 2. Close-up of Virtual Cockpit Screen

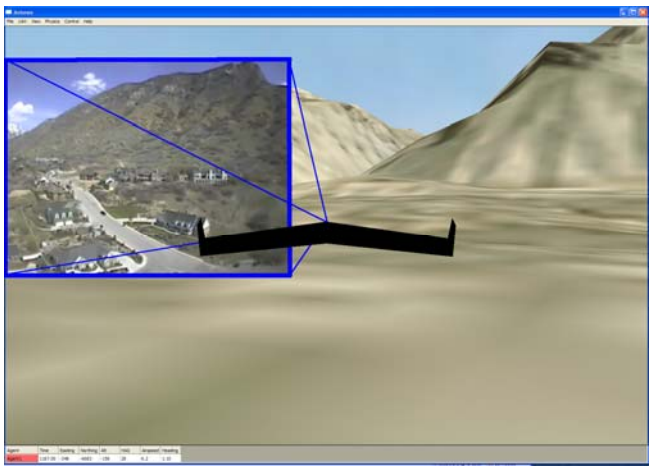

Figure 3. Screenshot of Augmented Virtuality Interface showing the outline of the aircraft in the center of the display and live video streaming into a subwindow

Table 4 contains a portion of the results of using the VGBF to characterize and compare the two interfaces. The table entries are keyed to the numbering system used in Table 1 and the paragraphs describing the VGBF in section 3. While the complete analysis cannot be presented here due to space limitations, the fragment in Table 4 illustrates the level of detail needed to describe the characteristics of each interface.

Table 4 also provides examples of characteristics that are similar as well as dissimilar for the two interfaces. The biggest similarity is that the two interfaces use the same output devices. An important difference is in their handling of the primary animated graphical visual output. While the Virtual Cockpit uses an unattached camera for a top-down view of navigation waypoints, the AVI uses external attachment to the primary object (the UAV avatar) to provide a visualization of the aircraft's streaming video and nearby terrain. In addition, the characterization brings out the fact that narrative integration is used much more in the AVI interface than the Virtual Cockpit.

5. DISCUSSION

As stated in the introduction, we see two uses of the VGBF: to evaluate existing interface designs (including comparing designs), and informing new design.

An advantage of using the VGBF for evaluating and/or comparing interfaces is that its use enforces a systematic look at each interface's characteristics. A closely-related disadvantage is that the results appear to give equal emphasis to all components, even though some interface elements may be more important than others when tailoring an interface for a particular user group.

The VGBF can inform HRI design by providing non-application-specific descriptions of a wide range of interface components commonly found in video games. We envision HRI designers using the framework to ask questions such as "would our intended users benefit most from an internally or externally attached camera?" (Note that we have developed draft guidance for which types of interface components may be best suited for certain situations, but describing this guidance is beyond the scope of this paper.) Once design choices have been made, designers can talk about their approaches with more precision. Rather than stating, "our interface was inspired by video games because it is similar to a FPS approach," designers can describe their interface's relationship to video games by using the specific language and detail of the VGBF.

A potential limitation of the VGBF is the fact that it is a reflection of the current state of the art of video games, and video game design advances are made rapidly. We have generalized the framework as much as possible with the goal of it also being applicable to the next generation(s?) of video game interfaces. As new interface components are developed, however, we believe the VGBF should be treated as a living framework and augmented with these components or amended to accommodate new developments.

Table 4. Fragment of Comparison of Two Interfaces Using VGBF

Category	Procerus Virtual Cockpit	Augmented Virtuality Interface (AVI)
	3.1 Output Devices	
3.1.1 Video screens	Video screen of a laptop computer. Used for all displays.	Video screen of a laptop computer. Used for all displays.
3.1.2 Speakers	Built in speakers of laptop computer or headphones plugged into laptop.	Built in speakers of laptop computer or headphones plugged into laptop.
3.1.3 Haptic devices	None.	None.
	3.2 Output Methods	
3.2.1 Primary animated graphical visual output		
3.2.1.1 Camera Perspective	Waypoint map display: Unattached camera focused on the map, top-down view. Artificial horizon display: Internal to an implied primary object representing aircraft cockpit.	External to the primary object (UAV avatar), above and behind.
3.2.1.2 Camera Movement	Waypoint map display: Top-down view, view plane fixed to the plane of the 2D map, view-up fixed to point N on map. VRP free to move through panning across the map in 2D and zooming in and out. Artificial horizon: Completely fixed to the implied primary object. Horizon indicator moves within this camera's field of view, showing movement of the UAV within the world.	"Chase plane perspective", VPN usually fixed to the primary object. Free camera can be invoked, but camera still moves in concert with motion of the primary object.
3.2.2 Additional Visual Output		
3.2.2.1.1 Type: Resources	Battery life, communication level, satellite coverage.	Battery life, communication level.
3.2.2.1.2 Type: Environmental Information	Location and orientation of aircraft within environment.	Location and orientation of aircraft within environment, orientation of visual sensor in relation to aircraft.
3.2.2.1.3 Type: Status	Current operating mode, current control mode, current navigation status, many aircraft parameters.	Aircraft parameters such as airspeed and altitude.
3.2.2.2.1 Location: Heads-Up Display	On the Artificial Horizon display, in the corners. Used for many types of information.	No heads-up display.
3.2.2.2.2 Location: Subinterfaces	Split into many tables, panes, panels.	Status info in a table attached outside of main display window. Streaming video on top of map
3.2.2.2.3 Location: Narrative Integration	Heading and location of aircraft on map, orientation of aircraft in Artificial Horizon.	Orientation and location of aircraft in 3D space against background terrain. Direction and position of visual sensor in relation to aircraft.
3.2.2.3 Temporality	Selectable displays in the same location, only one showing at a time. HUD on Artificial Horizon always on; displays info for currently selected aircraft. Summary table for all aircraft always on. Pop-up textual displays for alerts.	UAV avatar/terrain display and video subwindow always on, status of critical aircraft parameters at bottom of screen always on.
3.2.3 Non-Visual Output		
3.2.3.1.1 Audio: Sound Effect	System beep on errors and warnings.	None.
3.2.3.1.2 Audio: Music	None.	Musical theme on UAV launch.
3.2.3.1.3 Audio: Voice	None.	Low altitude and velocity warnings.
3.2.3.2.1 Haptic: Active Resistance	None.	None.
3.2.3.2.2 Haptic: Passive resistance	Passive resistance in joysticks and keyboard buttons. RC controller layout familiar to RC pilots, standard laptop keyboard and mouse.	Passive resistance in keyboard buttons. Standard laptop keyboard and mouse, familiar command button layout for emulated digital joysticks.
3.2.3.2.3 Haptic: Vibration	None.	None.

6. CONCLUSIONS

Based on a broad survey of video game interfaces, we have developed a set of domain-independent components to characterize, evaluate, and inspire HRI design. Although derived from video game interfaces, the VGBF itself is not specific to video games. We have illustrated the applicability of the framework to HRI by including numerous examples of components that are, or could be, in UAV interfaces.

As an example of the utility of the VGBF, we compared two UAV ground station implementations and presented a portion of the results in this paper. Using the VGBF encouraged a thorough examination of the interfaces and the results highlighted their similarities and differences. We feel other researchers, designers, and evaluators of HRI could use the VGBF to inspire new HRI design approaches and evaluate and compare existing designs.

7. ACKNOWLEDGMENTS

This work was supported by the United States Air Force Electronic Systems Center and performed under MITRE Mission Oriented Investigation and Experimentation (MOIE) Project 03057531 of contract 19628-94-C0001. All product names, trademarks, and registered trademarks are the property of their respective holders.

REFERENCES

[1] Chao, D. L. Doom as an interface for process management. In *Proceedings of the CHI 2001 Conference on Human Factors in Computing Systems.* ACM: Seattle, April 2001.

[2] Foley, van Dam, Feiner, and Hughes. Computer Graphics: Principles and Practice. Boston: Addison Wesley, 1990.

[3] Jones, H. L. and Snyder, M. Supervisory Control of Multiple Robots based on a Real-Time Strategy Game Interaction Paradigm. In *Proceedings of the 2001 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE: Tuscon, AZ, October 2001.

[4] Jørgensen, A. H. Marrying HCI/Usability and computer Games: A Preliminary Look. In *Proceedings of NordiCHI '04*. ACM: Tampere, Finland, October 2004.

[5] Lewis, M., Sycara, K., and Nourbakhsh, I. Developing a Testbed for Studying Human-Robot Interaction in Urban Search and Rescue. In Proceedings of the 10th International Conference on HCI. Crete, Greece, June 2003.

[6] Maxwell, B. A., Ward, N. and Heckel, F. Game-Based Design of Human-Robot Interfaces for Urban Search and Rescue. In *Proceedings of the CHI 2004 Conference on Human Factors in Computing Systems*. ACM: The Hague, April 2004.

[7] Pausch, R., Gold, R., Skell, T., and Thiel, D. What HCI designers can learn from video game designers. In Proceedings of the CHI 94 Conference on Human Factors in Computing Systems. ACM: Boston, April 1994.

[8] Tejada, S., Normand, E., and Sharma, S. Virtual Synergy: An Interface for Human-Robot-Agent Interaction. In *Proceedings of the AAMAS '03 Conference*. ACM: Melbourne, Australia, July 2003.

[9] Tvaryanas, A. P., Thompson, B. T., and Constable, S. H. U.S. Military Mishaps: Assessment of the Role of Human Factors Using HFACS. In *Proceedings of the 2nd Workshop on Human Factors of UAVs*. Cognitive Engineering Research Institute: Mesa, AZ, May 2005.

[10] Wolf, M. J. P. Genre and the video game. In *The Medium of the Video Game*, M. J. P. Wolf, ed. Austin: University of Texas Press, 2001. Also available at http://www.robinlionheart.com/gamedev/genres.xhtml

[11] Ye, Z. Genres as a tool for understanding and analyzing user experience in games. In Proceedings of the CHI 2004 Conference on Human Factors in Computing Systems. ACM: Vienna, Austria, April 2004.

VIDEO GAME REFERENCES

Advance Wars, Intelligent Systems Co. Ltd., Nintendo of America Inc., 2001

Alone in the Dark, Infogrames, Atari Europe S.A.S.U., 1992

Black & White, Lionhead Studios Ltd., Electronic Arts Inc., 2001

Deus Ex, Ion Storm Inc., Eidos Interactive Inc., 2000

Donkey Konga, Namco Limited, Nintendo Co. Ltd., 2003

Eternal Darkness: Sanity's Requiem, Silicon Knights Inc., Nintendo of Canada Ltd., 2002

Eye Toy: Play, SCEE Studio London, Sony Computer Entertainment Europe Ltd., 2003

Final Fantasy: Crystal Chronicles, The Game Designers Studio, Nintendo Co. Ltd., 2003

Half-Life 2, Valve Corporation, Vivendi Universal Games Inc., 2004

Halo 2, Bungie Studios, Microsoft Game Studios, 2004

Ico, SCEI, SCEA, 2001

Katamari Damacy, Namco Limited, Namco Limited, 2004

Nintendogs, Nintendo Co. Ltd., Nintendo Co. Ltd., 2005

Pikmin, Nintendo EAD, Nintendo Co. Ltd., 2001

Sid Meier's Civilization II, MicroProse Software Inc., MicroProse Software Inc., 1996

Super Mario Sunshine, Nintendo EAD, Nintendo Co. Ltd., 2002

Super Mario World, Nintendo Co. Ltd., Nintendo Co. Ltd., 1990

The Legend of Zelda: Ocarina of Time, Nintendo EAD, Nintendo Co. Ltd., 1998

The Legend of Zelda: The Wind Waker, Nintendo EAD, Nintendo Co. Ltd., 2002

Unreal Tournament, Epic Games Inc., GT Interactive Software Corp., 1999

URU: Ages Beyond Myst, Cyan Worlds Inc., Ubisoft Entertainment, 2003

WarioWare Touched!, Intelligent Systems Co. Ltd., Nintendo R&D1, Nintendo Co. Ltd., 2004