A Video Game Selection Process for Supporting Modding Projects

Eric Boivin, Francois Bernier

Defence Research & Development Canada – Valcartier

Stéphane Bouchard

Université du Québec en Outaouais

Contact: eric.boivin@drdc-rddc.gc.ca

Abstract

Most research & development (R&D) projects lack the financial resources to develop sophisticated VG-based training applications. Building on an existing VG resulting from a large development budget is a more frugal option. Although there are numerous examples of exploitation of VG in R&D projects, the selection process of these VGs and the possible modifications were rarely documented. First, this paper enumerates and classifies the possible forms of modifications and identifies the techniques for implementing these modifications in VGs. Then, it provides an example of a rigorous VG selection process for a R&D project on military training. The methodology requires identifying a set of VG titles that complied with a main criterion and then assesses them with other relevant criteria in order to determine the best candidate. Finally, this paper presents the architecture of a VG-based training system called *ImPACT* which implements a bio-feedback-driven gameplay. Sufficient details are provided so that other R&D projects may exploit a similar methodology in the future.

Keywods: modding technique, modding form, mod.

A Video Game Selection Process for Supporting Modding Projects

Video games (VGs) technology has continuously benefited of massive amount of investments over the last twenty years, the result being a significant technological improvement (www.dfcint.com). Virtual environments developed by open source community or by the industry for the scientific community pale into insignificance beside commercial VGs in term of functionality, graphic quality and content. Most research & development (R&D) projects or proof of concepts that require such advanced features do not have sufficient budgets to develop them from scratch. Modifying existing VGs resulting from large development budgets is a more frugal option. For instance, titles such as Unreal Tournament 3 (UT3) (2007) (www.unrealtournament.com) or Half-Life 2 (2004) (orange.half-life2.com) sequels provide various modification capabilities that make them suitable for prototyping or R&D.

Although research teams possess some computer science knowledge, they often lack modding expertise that limits the exploitation potential of VGs. The risk of using an ad-hoc approach is high with the consequence being a lack of rigour. For instance, VGs that correspond to the theme studied could be rejected based on a false assessment of their modding potential. Another danger is to select familiar titles without undertaking a proper comparative approach. This article is indented for scientists and software development teams with generic programming background that desire to exploit VGs to conduct training or experiments. Guidelines to understand programming requirements and various forms and techniques of modding are provided. In summary, how VG should be selected and how can it be modified are the two important questions that this paper tries to answer.

This paper is divided into three parts. Part ONE presents possible options for modifying a VG. Part TWO introduces the methodology and applies it to a case study. This process identifies the VG title on which a *Mod*[1]was developed. Part THREE presents the resulting architecture.

Part ONE: Understanding the potential of modding.

An overview of modding capabilities is necessary to better assess the modding potential a VG has to offer. Part ONE begins by enumerating and classifying possible forms of modification (modding forms) and then identifies techniques for implementing these modifications in VGs (modding techniques).

Modding forms

Inspired from Scacchi (2010), four forms of modding were retained and described under : (1) Graphical user interface (GUI) customization; (2) Content customization; (3) I/O customization and (4) Immersive customization.

GUI customization. This common form adapts the GUI in order to optimize the player efficiency during the game experience. The three most common GUI changes are (1) the development of better information representations (Figure 1), (2) the addition of a head-up display (HUD) or overlay over the scene to support the realism of environmental constrains (Figure 2), and (3) the reorganization of widgets.

[1] *Modding* and *Mod* refer to the action of modifying a VG and the modified VG itself respectively.

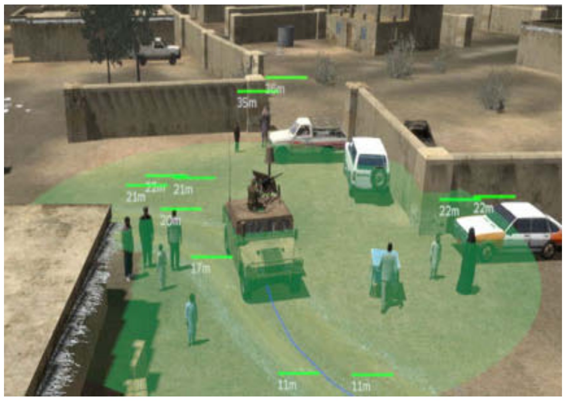
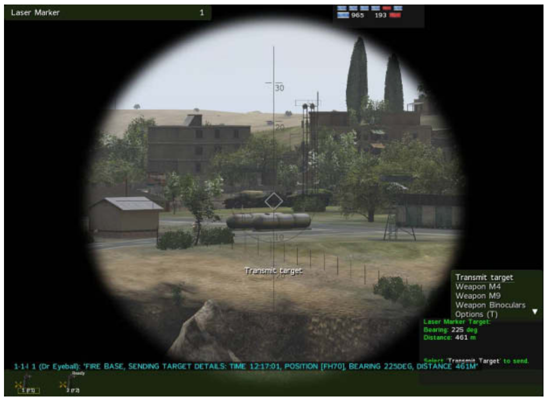


Figure 1: Example of better information representation extracted from VBS2 (www.vbs2.com).



1-1-1 1 (Dr Eyeball): 'FIRE BASE, SENDING TARGET DETAILS: TIME 12:17:01, POSITION [FH70], BEARING 225DEG, DISTANCE 461M'

Figure 2: Example of HUD extracted from VBS2 Fires.

The customization of external VG components, such as the mission briefing or after-action review summary, is considered as an extended form of GUI customization. These components provide user with relevant information both before and after the gameplay experience.

Content customization. Content customization modifies one or many aspects of the VG contents such as in-game objects, terrains, textures, animations, lighting, sound, music, camera settings, artificial intelligence rules, scripted actions, gameplay rules, levels, maps, campaigns, shaders, etc. For instance, content customization inspired the Canadian Forces Army Learning Support Centre (www.armyelearning.ca) in the development of the VG-based training system Canadian Forces: Direct Action (2008), a mod of the popular VG S.W.A.T.4 (2005), where police officer textures were replaced by military ones. VG community (e.g. ModDB website (www.moddb.com)) provides numerous examples of modified VGs.

I/O customization. I/O customization exploits external data to enhance game experience. This form was used by Dekker & Champion (2007) to develop a biofeedback-driven VG, where bio-sensing modifies the gameplay in real-time. Besides, I/O customization can support experimental analysis and after-action reviews sessions by tracking players' actions. Collected data helps to detect emergent problems, perform experimental analysis, and create a better experience. For instance, the level of difficulty of a map could be adjusted by analyzing length of typical VG sessions.

Immersive customization. This form emphasizes on enhancing a player's sense of presence. Stereoscopic visualization, multiple displays and 3D surrounds sound are examples. Some products can adapt VGs to Virtual Reality (VR) displays. For instance, 3D Vision (www.nvidia.com) enables stereoscopic visualization for up to three (stereoscopic) displays. CaveUT (Jacobson & Lewis, 2005) and getReal3D (www.mechdyne.com) libraries succeeded in implementing this form of customization for advanced immersive systems exploiting off-axis-rendering.

Figure 3 summarizes modding forms by representing them in a generic VG architecture.

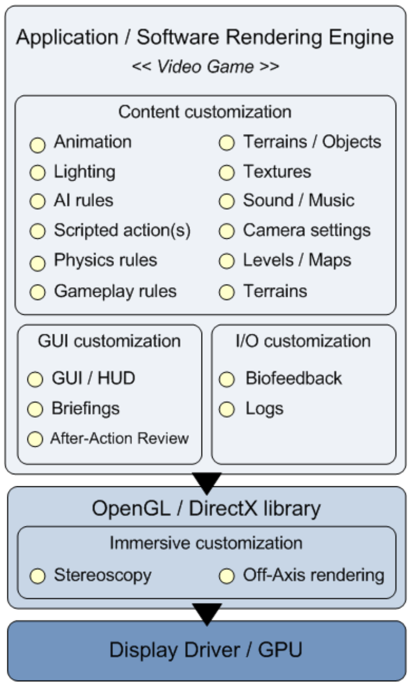


Figure 3: Distribution of modding forms into a generic VG architecture.

Modding techniques

VG must support the required specific functionalities to allow modding. Moreover, skilled programming personnel are required, especially when the documentation provided by the

VG developer is insufficient or simply inexistent. A skilled workforce supported by proper "modding" guidelines can overcome programming challenges or limitations.

For instance, customizing the GUI could be done by modifying scripts files on a given VG while it would require programming an external component on another one, with the same result for the player. Seven common approaches, called modding techniques, are described below: (1) Level editor, (2) Software development kits (SDK), (3) Scripting, (4) In-game command line console, (5) Configuration files, (6) File replacement and (7) Wrapper.

Level editor. A level editor is a GUI software application able to design or modify levels, maps, and campaigns of VGs. Level editors assemble 3D objects together in order to create complex scenes to which apply behaviors and events. Level editors are sometimes included within the VG distribution or can be developed by a game community. In general, level editors are easy to use but mainly devoted to the graphical aspect of game content. For instance, UT3, Skyrim: The Elder Scrolls V (2011) (www.elderscrolls.com/skyrim) and Gran Thieft Auto IV (2008) (www.rockstargames.com/IV) supply VG editor tools to assist mod makers in their customizations.

SDK and Scripting. SDKs and scripting techniques (e.g. UnrealScript (www.unrealengine.com/features/unrealscript)) offer the highest customization potential. Depending on the openness of the VG architecture, these techniques allow changing or enhancing VGs significantly. Since both approaches require programming and sometimes compilation, they offer more modding capabilities when they can be linked with third-party software. Overall, scripting is slightly less restrictive than SDK, even though there are scripting approaches offering high level of customization.

In-game command line console. This technique allows parameterizing of VG variables while the VG is being executed. Changes are made through a command line interface embedded into the VG. Examples of parameterization include VG avatars (e.g. speed, attack capabilities, stamina, artificial intelligence, etc.), VG environment (e.g.: lighting, seasons, physics behaviours, etc.), player experience (e.g.: HUD, VG gameplay rules, etc.) and computer system (e.g.: video resolutions, sound levels, network settings, etc.). In game command line console is often used for validating and testing effects of variable changes during the design process. Thereafter, retained changes are captured into VG configuration files. This kind of console could be found in Left 4 Dead (L4D) (2008) (left4dead.wikia.com/wiki/Console_commands).

Configuration files. This technique allows parameterizing variables prior to the VG is launched. Changes are made by modifying scripted files included into the VG distribution. These variables are usually related to GUI and VG content.

File replacement. File replacement technique consists in modifying or replacing files provided in the VG distribution, including 3D model files, textures files and sound files. It requires known and editable file formats, usually achieved via dedicated editors.

Wrapper. This modding technique consists in intercepting any function calls to a specific software dynamic library by redefining the original version; when an application starts, a wrapper library is loaded instead of the original one. This modding technique can extend existing functionalities, including those related to sound and rendering. For instance, it could be used to add a post-rendering HUD to display debugging information, modify projection matrices in real-time to apply stereoscopic effect or manage multi-display rendering. The wrapper is exploited in VRGL and CaveUT libraries (publicvr.org). This technique has the advantage of not requiring any modifications to the VG.

Table 1 summarizes the seven modding techniques described above.

Table 1: Modding techniques.

Modding techniques	**Description**	**Associated customization forms**	**Examples**
Level editor	GUI-based development.	VG content customization (especially for graphics aspects).	Animation, Levels / Maps, etc.
Software development kit	Various object-oriented programming forms (C++, C# ...) scripting forms (Phyton, LUA ...) or custom languages.	All customization forms.	GUI, Gameplay rules, Logs, etc.
Scripting			
In-game command line console	Functions called through a command shell interface embedded into the VG.	All customization forms (exclusively for existing VG settings and variables).	
Configuration files	Linear programming forms embedded in files.		
File replacement	Modification or replacement of original file contents.	GUI & VG content customization forms (done offline).	Terrains / Objects, Sound / Music, Textures, etc.
Wrapper	Interception of function calls by creating software component that minics the the original one.	Immersive & I/O customization forms.	Stereoscopy, Off-Axis rendering, Heads-up displays, etc.

Since there is no clear boundary between each modding technique neither a clear definition of what a level editor should do or not, it is difficult to identify the most adequate technique to achieve a customization form. A good understanding of the VG architecture capabilities is essential before committing time and resources to any modding techniques.

Finally, Table 2 summarizes the potential modding forms achievable by each modding technique. It is based on a review of VG architectures (Sheldon et al., 2009).

Table 2: Matrix of modding forms vs. modding techniques.

	Modding forms			
Modding techniques	**GUI customization**	**VG content customization**	**Data flow customization**	**Immersion customization**
Level editor		**G** - b		
Software development kit	**X**	g - **B**	**X**	**X**
Scripting	x	g - **B**	x	x
In-game command line console	x	g - b	x	x
Configuration files	x	g - b	x	
File replacement	x	g		
Wrapper	x (HUD)		x	**X**

Legend:
Bold upper case: strong modding potential
Normal lower case: limited modding potential
G/g: graphics aspects
B/b: behaviours aspect
X/x: whole aspects

Part TWO: Identifying a suitable VG title

A selection process is suggested to identify the most suitable title to develop a mod. First, a set of criteria is identified according to the project requirements. Each criterion is assessed and brought together to rank VG titles. The selection criterion considered the most difficult to fulfill should be assessed first in order to identify an initial set of VG to consider. The specificity of the scenario or a limited access to skilled developers often influences the choice of this criterion. Then, the modding potential (described in part ONE) is assessed for each VG titles. In addition to being itself considered as a selection criterion, the modding potential criterion could enhance the assessment of other criteria that do not fully meet project requirements.

The proposed methodology was applied to a military research project and is presented as an example. The Stress Management Training (SMT) project (Bernier, Boivin, & Bouchard, 2012), conducted in DRDC Valcartier, aimed at investigating the potential of VGs at practicing tactical breathing exercises, so that soldiers could be better prepared to cope with stress The exploitation of VG-based training exercises is an interesting solution to the problems of soldiers not buying-in and being non-compliant with the practice of SMT (Bouchard, Bernier, & Boivin, 2011; Stetz et al., 2007; Thompson & McCreary, 2006). Since a significant percentage of soldiers plays VGs (Orvis, Moore, Belanich, Murphy, & Horn, 2010), this technology may be well accepted by this population. In addition, the fact that these technologies have repeatedly demonstrated their effectiveness in similar contexts like the acquisition of military skills (Hays, 2005; Langkamer Ratwani, Orvis, & Knerr, 2010; Nullmeyer, Spiker, Golas, Logan, & Clemons, 2006; Roman & Brown, 2008) strengthen the confidence in the proposed approach. Using an existing VG was considered the only available option.

Part TWO applies the VG selection process to SMT mod development. The assessment was based on five selection criteria. The selection of criteria focused first on identifying a set of stressful VG titles that allow trainees to practice tactical breathing. Adaptable, Ethical, Experiment-friendly and Immersive criteria were successively identified and assessed to determine the best candidate. Each criterion was rated on a scale ranging from 0 (least) to 5 (most). The Adaptable criterion, which determines the modding potential, must be handled carefully because VGs that scored high on this criterion can be changed to improve other criterion that scores poorly. Note that modding potential of the most promising VG candidates should be further analyzed by developing small prototypes in order to determine the “best fit”. Prototyping confirms modding expectations and helps to detect potential issues such as computational and programming limitations.

(1) Stressful criterion

Preliminary investigations ruled early that the required training scenario should be stressful enough to allow tactical breathing practice. Two options were identified. The first one was to develop a tailored scenario for an existing VG-based training system such as Virtual Battlespace 2[2] (2007) (VBS2). Although VBS2 has sophisticated authoring tools, this option was set aside because of the difficulty to develop a convincing stressful scenario with limited resources. Developing a stressful VG scenario is a complex task (Perron, 2004). It requires art-style skills and fine tuning. The second option was to adapt a VG that already implements a stressful scenario. The survivor horror genre was identified as the most stressful of all game genres. This choice was based on the fact that it was less risky to add missing functionalities (e.g. experimental needs) instead of taking the risk of developing an unsuitable scenario.

The evaluation of VG titles was based on the subjective ranking of six “top-10 scariest video games” done by six websites. Each entry was worth 1 point over a maximum of 5. It should be noted that the overall ranking of scariest games is probably inflated by the popularity of some titles. Nevertheless, it was considered the best approach in the absence of a more objective alternative. Table 3 shows a list of 8 stressful titles initially identified. This set was retained for the remaining criterion assessments. PC-based VGs only were considered, for console-based VGs could not be modified.

[2] VBS2 is a virtual tactical training system designed by *Bohemia Interactive Studio*.

Table 3 Video games titles reviewed: Stressful criterion.

	Websites						
Titles (PC-based & Survivor horror styles)	About[A] (2009)	Ask Men[B] (2010)	Bright Hub[C] (2010)	Gossip Gamers [D] (2009)	Joystick Division[E] (2010)	Pop Crunch[F] (2010)	**Total**
Condemned: Criminal Origins (2006)	-	1	-	1	1	-	3
Dead Space (2009)	1	1	1	-	1	1	5
F.E.A.R. (2005)	1	-	1	1	1	1	5
F.E.A.R. 2 (2009)	1	1	-	-	-	-	2
Killing Floor (2009)	1	-	-	-	-	-	1
Left 4 Dead (2008)	1	1	1	-	-	-	3
Penumbra: Black Plague (2008)	1	-	1	-	-	-	2
Resident Evil 4 (2005)	1	1	-	-	-	-	2

Notes

A – http://compactiongames.about.com/od/topgames/tp/scarygames.htm.

B – http://www.askmen.com/top_10/entertainment/top-10-scariest-video-games.html.

C – http://www.brighthub.com/video-games/pc/articles/46528.aspx.

D – http://www.gossipgamers.com/top-10-scariest-games-of-all-time.

E – http://www.joystickdivision.com/2010/02/top_ten_scariest_games_of_all.php.

F – http://www.popcrunch.com/the-13-scariest-video-games-of-all-time.

(2) Adaptable criterion

The aforementioned modding techniques were assessed for each VG title. Table 4 shows the results of assessment where one point was given for each supported modding technique.

Table 4: Video games titles reviewed: Adaptable criterion.

	Modding capabilities						
Titles	Level Editor (+1)	SDK (+1)	Scripting (+1)	In-game console or Config files (+1)	File replacement (+1)	Wrapper (+1)	**Total**
Condemned: Criminal Origins (2006)	-	-	-	-	-	-	**0**
Dead Space (2009)	-	-	-	-	-	-	**0**
F.E.A.R. (2005)	1	1	1	1	1	1	**6(5)**
F.E.A.R. 2 (2009)	-	-	-	-	-	-	**0**
Killing Floor (2009)	1	-	1	1	1	1	**5**
Left 4 Dead (2008)	1	1	1	1	1	1	**6(5)**
Penumbra: Black Plague[A] (2008)	-	-	-	-	1	-	**1**
Resident Evil 4 (2005)	-	-	-	-	1	-	**1**

Notes:

A – The first of the sequel, Penumbra: Overture (2007), is now available in open source on demand.

Adaptability provides many advantages. An adaptable VG can have a positive effect on others criterion assessments by enhancing or correcting weakness associated to them.

(3) Ethical criterion

The survivor horror VG genre draws on horror fiction to provide stressful scenario. Therefore, some VGs could present unsuitable content referring to violence, gore, nudity and/or drugs. This constraint was taken into account into this criterion assessment in order to foster the selection of VG titles free of ethical issue and inappropriate content.

The assessment was inspired from the Entertainment Software Rating Board (ESRB, esrb.org). Five points were initially allocated to each VG title because of their Mature rating. VG involving strong language, alcohol and drugs, very intense violence with blood and gore, or sexual themes were penalized (i.e. loss of 1 point) while games involving some violence, conflict, weapons and injuries were tolerated. Table 5 shows the assessment grid for this criterion. Some penalties were excluded because they could be removed by modifying the VG.

Table 5: Video games titles reviewed: Ethical criterion.

	ESRB Initial Rating[A]	**ESRB Detailed Rating**					
Titles	Mature (+5)	Language (0) or Strong Language (-1)	Use of Drugs and Alcohol (-1)	Violence (0) or Intense Violence (-1)	Blood (0) or Blood and Gore (-1)	Partial Nudity (0) or Sexual Theme (-1)	**Total**
Condemned: Criminal Origins (2006)	5	-1	-	-1	-1	-	**2**
Dead Space (2009)	5	-1	-	-1	-1	-	**3**
F.E.A.R. (2005)	5	-1	-	-1	-1	-	**2**
F.E.A.R. 2 (2009)	5	-1	-	-1	-1	-1	**1**
Killing Floor (2009)	5	-1[B]	-	-1	-1	0[B]	**3**
Left 4 Dead (2008)	5	0[B]	-	-1	-1	-	**3**
Penumbra: Black Plague (2008)	5	0	-	0	-1	-	**4**
Resident Evil 4 (2005)	5	-1	-	-1	-1	-	**2**

Notes:

A – Entertainment Software Rating Board – http://www.esrb.org.

B – Excluded from assessment. These elements could be removed by modifying the VG content (see Adaptable criterion).

(4) Experiment-“friendly” criterion

Typically, VG titles are not intended for supporting training and experiments. Some VG features associated to the point of view (POV) of the player, the complexity of the storyline, and the variability of player actions should be carefully assessed in order to ensure short experiment sessions. Six features were identified and assessed (Table 6).

Table 6: Video games titles reviewed: Experiment-"friendly" criterion.

		POV		Storyline & Actions				
Titles		First Person (+3)	Third Person (+2)	Short intro. (+1)	Linear prog. (+1)	Adventure (-1)	Puzzle-solving	**Total**
Condemned: Criminal Origins	(2006)	3	-	1	1	-1	-1	**3**
Dead Space	(2009)	-	2	-	1	-	-	**3**
F.E.A.R.	(2005)	3	-	-	1	-	-	**4**
F.E.A.R. 2	(2009)	3	-	-	1	-	-	**4**
Killing Floor	(2009)	3	-	1	-	-	-	**4**
Left 4 Dead	(2008)	3	-	1	1	-	-	**5**
Penumbra: Black Plague	(2008)	3	-	-	-	-1	-1	**1**
Resident Evil 4	(2005)	-	2	-	1	-	-	**3**

A first-person POV was identified as being more suitable for experiments. It maximizes the involvement and the immersion of players by experiencing the action through the eyes of an avatar. A third-person POV was considered acceptable with a lower score. Experimental friendliness included the possibility to minimize the experimentation time. It was decided that participants would be trained in the VG for a maximum of 20 minutes. Consequently, the VG had to avoid long introductory phases. In addition, linear storyline progression style was preferred in opposition of sandbox-style. A linear progression style offers less variability in term of player actions. Finally, VG titles having adventure and/or puzzle-solving styles were penalized because they could break the rhythm of gameplay and interfere with the training session.

(5) Immersive criterion

VR was identified as a must feature because it can be exploited to induce stress for practicing coping strategies. Robillard et al. (2003) found a strong relationship between anxiety and presence resulting from the use of immersive devices. The NVIDIA 3D Vision was identified as a simple solution to enable stereoscopy, without the need for special game patches. The rating chart proposed by NVIDIA was directly reused for the assessment of this criterion (Table 7).

Table 7: Video games titles reviewed: Immersive criterion

Titles	NVIDIA 3D Vision Kit	Total
Condemned: Criminal Origins (2006)	0[B]	**0**
Dead Space (2009)	2[A]	**2**
F.E.A.R. (2005)	4[A]	**4**
F.E.A.R. 2 (2009)	2[A]	**2**
Killing Floor (2009)	4[A]	**4**
Left 4 Dead (2008)	4[A]	**4**
Penumbra: Black Plague (2008)	0[B]	**0**
Resident Evil 4 (2005)	0[B]	**0**

Notes:

A – NVIDIA 3D vision rating (nvidia.com/3d-vision)
(Fully supported = 5, Excellent = 4, Good = 3, Fair = 2, Poor = 1 & Unsupported = 0)

B – Unrated

Compilation of assessments

Left 4 Dead (L4D) was selected for designing the mod among a group of three VGs that clearly stands out above the others (F.E.A.R., Killing Floor and L4D) (Table 8). Considering their similar assessments in terms of modding capabilities, L4D was preferred to Killing Floor because of its linear storyline and its graphical quality. Bouchard et al. (2011) confirmed the modding potential of L4D and revealed that it is sufficiently stressful and yet appreciated by soldiers as a training stressor. Finally, even if F.E.A.R subjectively appeared to be more stressful, it suffered of weaknesses in term of experiment-“friendly” (i.e. long introduction phase) and ethical criteria.

Table 8: Review of a selection of video games titles.

Titles	**Criterion**					
	Stressful	Adaptable	Ethical	Friendly-Exp.	Immersive	**Total**
Condemned: Criminal Origins (2006)	3	0	2	3	0	8
Dead Space (2009)	5	0	3	3	2	13
F.E.A.R. (2005)	**5**	**5**	**2**	**4**	**4**	**20**
F.E.A.R. 2 (2009)	2	0	1	4	2	9
Killing Floor (2009)	**1**	**5**	**3**	**4**	**4**	**17**
Left 4 Dead (2008)	**3**	**5**	**3**	**5**	**4**	**20**
Penumbra: Black Plague (2008)	2	1	4	1	0	8
Resident Evil 4 (2005)	2	1	2	3	0	8

Part THREE: Architecture of the Mod ImPACT

Once a VG is selected, a software development methodology must be exploited to ensure that the mod complies with the requirements of the application. Part THREE presents an example of the software development process used to develop the SMT mod, called ImPACT. It relies on the Unified Modeling Language (Booch et al., 2005), a standardized modeling language designed for object-oriented software engineering. The following sections present two diagrams (views) that were used to create and document the architecture: use case, and class diagrams.

Use case diagram

One of the objectives of *ImPACT* was to force the player to relax in order to achieve the mission. The player arousal had to be deduced from biofeedback sensors and exploited to motivate the practice of tactical breaking into a dynamic gameplay experience. Visual and audio feedbacks were added so that players could be informed of their level of stress. Finally, instructors required control mechanisms for tailoring the training session. The use case diagram, showed in Figure 4, captures the interaction between the mod and its users. A system operator was added to manage tasks related to set up and monitor the training session.

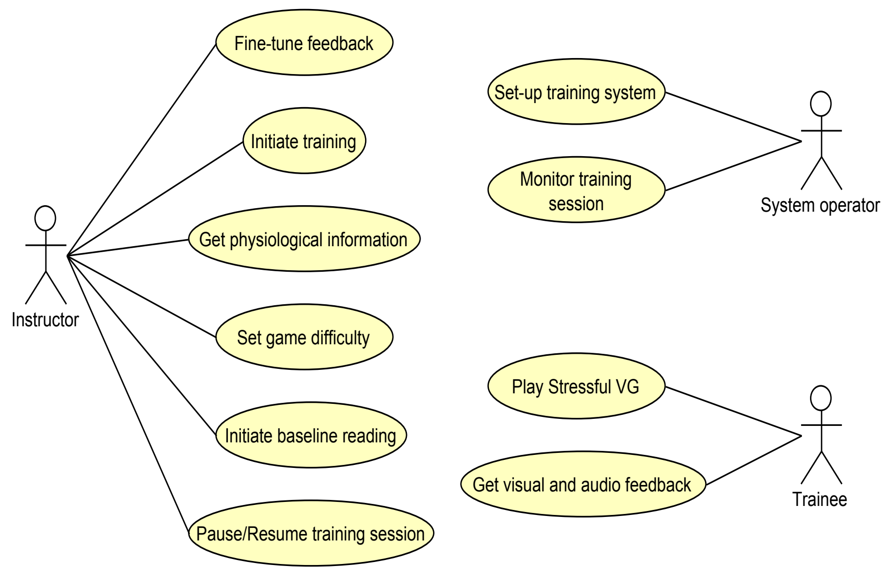


Figure 4: ImPACT use case diagram.

Class diagram

Class diagram aims a representing the structure of a system, including its main elements, the relationships between them and their grouping into packages. It shows how the original VG fits within the global application. The class diagram, showed in Figure 5, describes the structure of ImPACT in five main packages. Those packages can be classified into two categories.

A first group of packages that includes Controller, Physiological data acquisition, and Physiological data display, is external to the VG and to the computer where the VG runs. These packages are no less part of the mod and they interact with the VG via added modules through the network. The Physiological data acquisition package captures biometric information from sensors and encodes it for its use by other packages. The Physiological data display package shows the instructor current and past physiological data. Finally, the Controller package provides instructor with all functionalities described in the use case diagram, and determines the level of arousal of the player in real-time.

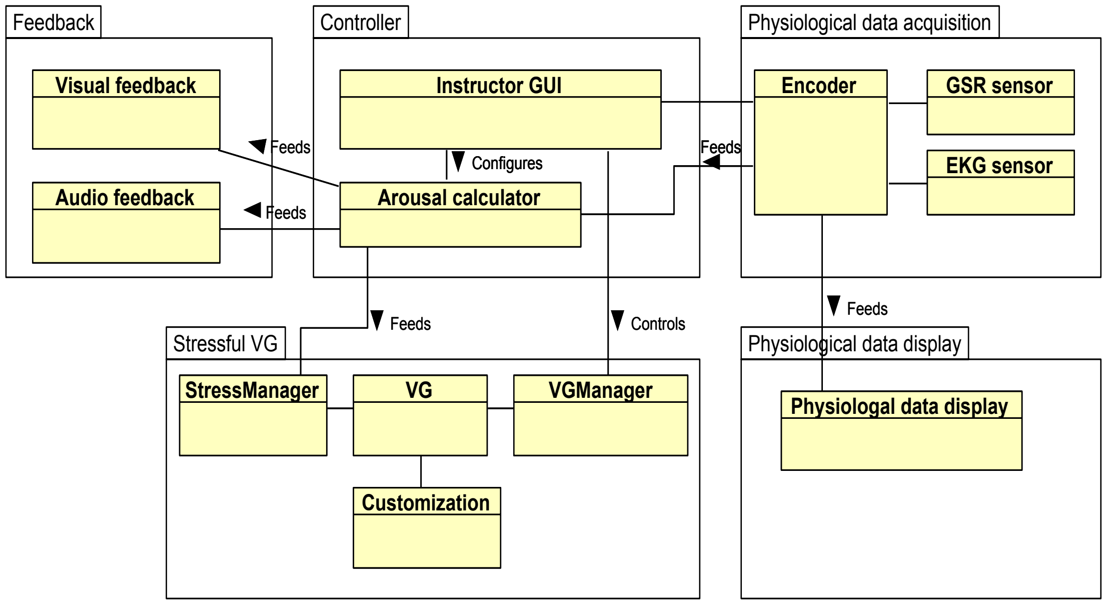


Figure 5: ImPACT class diagram.

The second group of packages comprises classes that are deployed within the VG itself and then are more constraint by the modding potential. The Stressful VG and Feedback packages were implemented by using various modding forms and techniques. First, the conversion to a biofeedback-driven gameplay supervised by an instructor was ensured by the SDK modding technique, which sets the VG difficulty, and by the Wrapper modding technique, which implements visual and audio feedbacks. Second, the Experiment-friendly criterion was adjusted by applying VG content modifications via Level editor, Configuration file and Replacement file modding techniques. These modifications narrowed the gamer options and indirectly helped to reduce required time for the familiarization phase and to generate datasets easier to analyze. The following paragraphs describe how each technique was implemented.

Feedback package. Two functionalities were required for this package. First, the field of view was adjusted accordingly with the physiological measures. Second, the sound of a heart audio feedback complemented the visual feedback by progressively increasing in rate and loudness also depending on the physiological measures. The Wrapper modding technique was used for achieving both functions giving the illusion that the visual and the audio feedbacks came from the VG itself.

Stressful VG package. VG represents the selected VG title, L4D. It objective is to stress, "entertain", captivate and provide a "mission" objective to the trainee. StressManager class adjusts the level of difficulty of the VG according to the trainee's level of arousal by using the L4D SDK with SourceMod (www.sourcemod.net). VGManager class executes the commands (start, pause, set difficulty level) received from the instructor by using the L4D SDK. Customization class adapts the VG content to the training needs. Modifications were applied to L4D configuration files to standardize the training experience among participants. Specific keyboard keys and widgets were disabled to restrict player’s actions. Initial player’s equipment was limited to a personal pistol and a riffle. Several objects (e.g. medic kit, bomb or molotov cocktail) were removed from the scenario by using Hammer, the L4D level editor (developer.valvesoftware.com). Mission pathway was slightly modified to constraint player’s displacements. Partially nude characters and strong language expressions were both disabled to correct ethical issues by using respectively the Configuration file and the File replacement techniques. Finally, L4D tracking mechanism was enabled through L4D configuration files to monitor player experience. For instance, player’s kills, spawned enemies and length of mission could be retrieved from logs. Table *9* summarizes applied modifications for ImPACT.

Table 9: Changes applied to L4D: Matrix of modding forms vs. modding techniques.

	Modding forms			
Modding techniques	**GUI customization**	**VG content customization**	**Data flow customization**	**Immersion customization**
Level editor (Hammer)		L4D Hospital level • Alternative pathways are removed; • Bonus items are removed.		
Software development kit (L4D SDK)			Stressful Manager class • Gamplay difficulty is adjusted according to the Arousal Calculator class; VG Manager class • Gamplay difficulty is adjusted by the Instructor GUI class.	
Scripting				
In-game command line console	The L4D In-game command-line console was used for validating and testing setting changes in real-time. Retained settings were programmed into L4D configuration files.			
Configuration files	L4D game settings • Game launching process is automated; • Introduction scene is bypassed; • HUD is lightened.	L4D game settings • Allocated Player's equipment is lightened; • Enemies' AI parameters and actions are modified.	L4D game settings • Monitoring and tracking functions are activated for post-analysis purposes.	
File replacement		L4D game settings • Strong language quotes are removed.		
Wrapper	Visual feddback • L4D HUD is overlayed by a tunneled vision effect. Opacity is determined by the player's stress level.			Visual feedback • Stereoscopic rendering is added. Audio Feedback • Emulated player's breathing sound is added. Sound level and rhythm are determined by the player's stress level.

The implementation was completed and two experiments were conducted (S. Bouchard, Bernier, Boivin, Guitard, et al., 2012; S. Bouchard, Bernier, Boivin, Morin, & Robillard, 2012) to assess the efficacy of ImPACT to train soldiers to cope with stress. Results have shown that, even with a mod as simple as the one presented here, soldiers trained with ImPACT mod were better at coping with stress. It is not possible to claim that the methodology employed to develop ImPACT was the cause of the positive results, but it nevertheless it reduced uncertainty and provided more options during the selection and implementation processes.

Conclusion

This paper tried to answer two questions: how VG should be selected and how can it be modified to support projects aiming to develop a mod. First, possible modding forms and techniques were presented. Second, a simple methodology was introduced for identifying the best VG title for modding. Finally, ImPACT was used as a case study for validating the suggested VG selection process.

There is no doubt that modding offers great opportunities for R&D activities. A few VG titles are enough flexible to undertake projects mainly focused on prototyping or proof of concept objectives. However, before involving resource into a modding project, a prior-investigation of the VG modding capabilities and application requirements should be done rigorously. Even if modding community provides useful insights on “HOW TO” programming questions, experienced developers are required, especially when the documentation is incomplete or simply inexistent.

The presented VG selection process helps to minimize the risk of choosing an unsuitable VG title for modding projects. Although the process was developed for SMT, it remains quite generic and is applicable to other modding projects. Good results were obtained for ImPACT, and this documented case could contribute to design a more generic and elaborated methodology

Acknowledgments

Portions of this paper were included in an internal research report presented by authors to the Canadian Forces (Bernier et al., 2012). The opinions expressed in this publication reflect those of the authors and do not necessarily represent the opinion of the Canadian Forces or the Department of National Defence.

References

Bernier, F., Boivin, E., & Bouchard, S. (2012). Teaching stress management skills to soldiers. DRDC Valcartier.

Booch, G., Rumbaugh, J., & Jacobson, I. (2005). *Unified Modeling Language User Guide, The (2nd Edition) (Addison-Wesley Object Technology Series)*. Addison-Wesley Professional.

Bouchard, S., Bernier, F., Boivin, E., Guitard, T., Laforest, M., Dumoulin, S., & Robillard, G. (2012). Modes of immersion and stress induced by commercial (off-the-shelf) 3D games. *Defense Modeling and Simulation: Applications, Methodology, Technology (JDMS)*.

Bouchard, S., Bernier, F., Boivin, E., Morin, B., & Robillard, G. (2012). Using Biofeedback While Immersed in a Stressful Videogame Increases the Effectiveness of Stress Management Skills in Soldiers. *PlosOne*, *7*(4).

Bouchard, Stéphane, Bernier, F., & Boivin, É. (2011). Initial steps in inducing stress to teach stress management skills to soldiers: Testing stressors, immersive technologies and avatars. DRDC Valcartier.

Dekker, A., & Champion, E. (2007). Please biofeed the zombies: enhancing the gameplay and display of a horror game using biofeedback. *Proc. of DiGRA*.

Hays, R. T. (2005). *The effectiveness of instructional games: A literature review and discussion*. DTIC Document.

Langkamer Ratwani, K., Orvis, K. L., & Knerr, B. W. (2010). *Game-Based Training Effectiveness Evaluation in an Operational Setting*. DTIC Document.

Nullmeyer, R. T., Spiker, V. A., Golas, K. C., Logan, R. C., & Clemons, L. (2006). The Effectiveness of a PC-Based C-130: Crew Resource Management Aircrew Training Device. *The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC)* (Vol. 2006).

Orvis, K. A., Moore, J. C., Belanich, J., Murphy, J. S., & Horn, D. B. (2010). Are soldiers gamers? videogame usage among soldiers and implications for the effective use of serious videogames for military training. *Military psychology*, *22*(2), 143–157.

Perron, B. (2004). Sign of a threat: The effects of warning systems in survival horror games. Proceedings of COSIGN 2004 (pp. 132–141). Split, Croatia.

Robillard, G., Bouchard, S., Fournier, T., & Renaud, P. (2003). Anxiety and presence during VR immersion: A comparative study of the reactions of phobic and non-phobic participants in therapeutic virtual environments derived from computer games. *CyberPsychology & Behavior*, *6*(5), 467–476.

Roman, P. A., & Brown, D. (2008). Games–Just how serious are they? *The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC)* (Vol. 2008).

Scacchi, W. (2010). Computer game mods, modders, modding, and the mod scene. *First Monday*, *15*(5).

Sheldon, A., Bolduc, F., Bouchard, M., Cocking, M., Courchesne, M., Funk, C., Guay, J.-S., et al. (2009). Revue du Serious Gaming. DRDC Valcatier (Wiki No. CR 2009-501). Québec:

Stetz, M. C., Thomas, M. L., Russo, M. B., Stetz, T. A., Wildzunas, R. M., McDonald, J. J., Wiederhold, B. K., et al. (2007). Stress, mental health, and cognition: a brief review of relationships and countermeasures. *Aviation, space, and environmental medicine*, *78*(Supplement 1), B252–B260.

Thompson, M. M., & McCreary, D. R. (2006). *Enhancing mental readiness in military personnel*. DTIC Document.