Copy No._____

Defence Research and Development Canada
Recherche et développement pour la défense Canada

DEFENCE R&D DÉFENSE

Development of a CPF Model in UnReal Tournament

J. Bennet, J. Frim and D. Tack
Humansystems Incorporated

Humansystems Incorporated
111 Farquhar St., 2nd floor
Guelph, Ontario N1H 3N4

Project Manager: Mark Hazen 902 426-3100 x176
Contract Number: W7707-4-2677/A
Contract Scientific Authority: Allan Gillis 902 426-3100 x203

Defence R&D Canada – Atlantic

Contract Report
DRDC Atlantic CR 2005-197
January 2006

Canada

This page intentionally left blank.

Development of a CPF Model in UnReal Tournament

J. Bennet
J. Frim
D. Tack

Prepared by:

HumanSystems
111 Farquhar St, 2nd Floor
Guelph, ON N1H 2N4

Project Manager: Mark G. Hazen, 902 426 3100 x176
Contract number: W7707-4-2677/A
Contract Scientific Authority: Allan Gillis 902 426 3100 x203

Defence R&D Canada – Atlantic

Contract Report
DRDC Atlantic CR 2005-197
January 2006

Principal Author

Original signed by J. Bennet

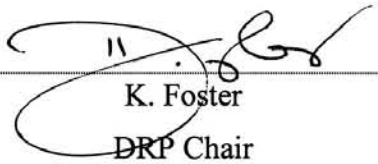
J. Bennet

Approved by

Mark Hazen

Scientific Authority

Approved for release by



K. Foster

DRP Chair

Work was completed as part of ARP 11BK

Abstract

A 3D model of a Canadian Patrol Frigate (CPF) was imported into a version of the UnReal Tournament multi-player game. This document gives details of the problems encountered and resolutions developed during the import process. The result of the project is a set of playable levels that represent the upper deck, hanger and bridge of a Halifax class frigate.

Résumé

Une modélisation 3D d'une Frégate canadienne de patrouille (FPC) a été importée dans une version du jeu à joueurs multiples UnReal Tournament. Le présent document donne le détail des problèmes rencontrés et les solutions élaborées pendant le processus d'importation. Le résultat du projet est un ensemble de niveaux jouables représentant le pont supérieur, le hangar et la passerelle d'une frégate de la classe Halifax.

This page intentionally left blank.

Executive summary

Development of a CPF Model in UnReal Tournament

Bennet, J., Frim, J. and Tack, D.; DRDC Atlantic CR 2005-197; Defence R&D Canada – Atlantic; January 2006.

Introduction

DRDC Atlantic is investigating the use of commercial off the shelf (COTS) games for use in concept development and experimentation (CD&E). One type that has been used in other countries to great effect are multiplayer First Person Shooter (FPS) games. In terms of naval operations FPS games seem well suited to the modelling of force protection scenarios. In order to assess this, a contract was let to evaluate FPS game engines and to build a FPS model of a Canadian Patrol Frigate (CPF) along side a dock.

Results

A CPF model allowing access to the deck, hanger and bridge areas was developed from blueprints and imported into the Unreal engine. In addition, the use of observers and virtual cameras to simulate surveillance assets were implemented. The importation and development of the model was not straight forward and many graphics issues had to be resolved.

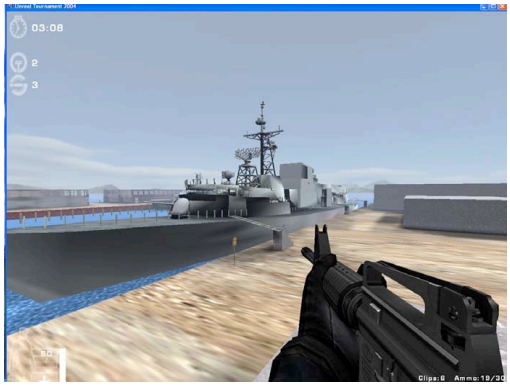


Figure 1: Final CPF Model in Unreal StrikeForce Mod.

Significance

It has been successfully demonstrated that a model of a Canadian ship can be imported into a FPS game to provide a virtual environment for force protection CD&E.

Future plans

It is planned to continue the development of such models to support the Force Protection research program. As such more detail is needed in the area of third party non-combatants, Canadian uniforms for players, the addition of Canadian Force Protection weapon systems etc. In addition other types of COTS games are being investigated for use by the Canadian Forces.

Sommaire

Développement de modélisation d'une FPC pour UnReal Tournament

Bennet, J., Frim, J. et Tack, D.; RDDC Atlantique CR 2005-197; R & D pour la défense Canada – Atlantique; Janvier 2006.

Introduction

RDDC Atlantique étudie la possibilité d'utiliser des jeux disponibles sur le marché (COTS) pour l'élaboration et l'expérimentation de concepts (EEC). Un des types utilisés avec grand succès par d'autres pays est le jeu de tir à joueurs multiples. En termes d'opérations navales, les jeux de tir semblent bien adaptés à la modélisation des scénarios de protection de la Force. Afin d'évaluer cette possibilité, un contrat a été attribué pour évaluer les moteurs de jeux de tir et élaborer un modèle de jeu avec une Frégate canadienne de patrouille (FPC) à quai.

Résultats

La modélisation d'une FPC permettant l'accès au pont, au hangar et à la passerelle a été développée à partir des plans d'un navire et importé dans le moteur Unreal. De plus, l'utilisation d'observateurs et de caméras virtuelles pour simuler le système de surveillance a été appliquée. L'importation et le développement de la modélisation n'ont pas été faciles et de nombreux problèmes graphiques ont dû être résolus

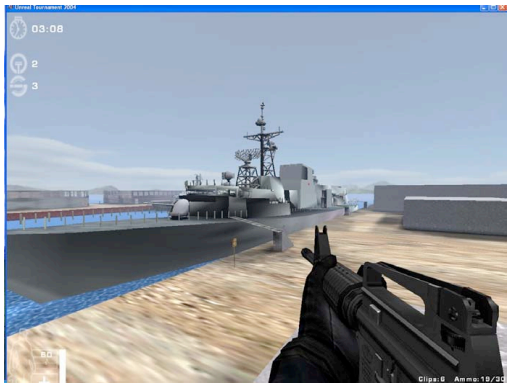


Figure 1: Modélisation finale de la FPC pour le jeu Unreal StrikeForce

Importance

Il a été démontré avec succès que la modélisation d'un navire canadien peut être importée dans un jeu de tir afin d'assurer un environnement virtuel pour l'EEC de protection de la Force.

Perspectives

Il est prévu de poursuivre le développement de tels modèles afin de soutenir le programme de recherche de protection de la Force. Pour cela, plus de détails sont nécessaires en ce qui a trait aux tiers non-combattants, aux uniformes canadiens pour les joueurs, à l'ajout de système

Table of contents

List of figures

1. Introduction

This Contractor Report summarizes the current status of the Force Protection Simulation Demonstration from a contract awarded to Human*systems*® Incorporated (HSI®) from Defence Research and Development Canada – Atlantic (DRDC Atlantic). This project investigated the feasibility of using a commercial first-person shooter (FPS) game to force protection scenarios on a Halifax class Canadian Patrol Frigate. In the first phase of the contract, five FPS game engines were evaluated. [1]

This contractor report covers the second phase of the contract in which a CPF was implemented in the chosen FPS game engine; UnReal Tournament [2]. The report is organized into the following sections.

1. Competed Work Summary

2. Software Issues and Resolutions

3. Recommended Way Ahead

Figure 1: Final CPF Model in Unreal StrikeForce Mod.

2. Completed Work Summary

The overall objective of the Force Protection Simulation Demonstration was to determine the feasibility of using a commercial first-person shooter game to simulate assault and defence scenarios of a warship while in port. Human*systems*® reviewed existing off-the-shelf commercial 3D game engines and developed a 3-dimensional model of a Halifax class Canadian Patrol frigate (including upper decks and some interior spaces) to provide some realism for the initial investigations.

The following section details the work completed to date on the Force Protection Simulation Demonstration project:

- Reviewed first-person 3D game engines for their suitability in developing the Force Protection defence simulation.
- Generated a skybox (cube comprised of six images) to provide a distant 3D backdrop for the dockyard environment.
- Identified the Strike Force mod for Unreal Tournament 2004. This provided a platform for realistic tactical simulation (assault weapons, uniforms, physics, etc.).
- Reconstructed structural components of the DRDC provided VRML frigate model using 3D Studio Max. These structural components included: the ship hull, bridge, and hangar. The bridge and hangar reconstruction included the addition of interior rooms, alcoves, and passageways to facilitate personnel movement and realistic defence scenarios. The hangar errors with the VRML frigate geometry were corrected.
- Other frigate components were extracted from the VRML model and transformed into individual static mesh objects (ASE format). The frigate model was then prepared to be imported successfully into UnrealEd.
- Created and packaged (UnReal Game Engine requirement) realistic, customized texture maps to enhance the CPF 3-dimensional model. The majority of the custom textures were created from images obtained from the February CPF photo shoot.
- Located various prefab static mesh models for the purpose of enhancing the CPF model and dockyard environment (e.g., pipes on frigate, shipping container for dock, etc.)
- Completed the original construction of Land Control Officer's station (including texture map) within 3D Studio Max and imported into UnrealEd.
- Other enhancements to the CPF model included: a guide wire added to upper deck perimeter posts to prevent players from inadvertently falling off frigate; zodiac static mesh textured and imported into CPF model; realistic textures applied to exterior of CPF and flight deck; construction of gangway from CPF to pier; flight deck netting texture created and added to flight deck of CPF; maple leaf logo added to frigate funnel; glass added to windows of bridge.
- Developed and installed security cameras and monitors feature. Four exterior security cameras can now capture and transmit video to four security monitors located on the CPF

bridge. One of the security cameras is mounted on a UAV. All networked players can view video transmission.

- Investigated the process required for skinning Attacking and Defending forces with appropriate uniform specifications (e.g., CADPAT). The actual skinning of the defensive force was not completed due to budget and time constraints.
- Created a small UAV (static mesh helicopter) that flies above the frigate after it has been activated by a game player (activation requires player to fire at UAV).
- Continued debugging and troubleshooting of Unreal Game Engine. Documented issues and resolutions (refer to section 3.0 “Software Issues and Resolutions”).
- Provided a Phase 1 demonstration consisting of Unreal Tournament 2004 installation CDs, StrikeForce game mod, static mesh model of CPF, 3D dockyard environment, and installation instructions.
- Delivered modified Force Protection simulation game files for DRDC Atlantic client demonstration purposes.
- Completed and delivered Phase 2 deliverables (i.e., Force Protection simulation game files and Technical Memo).
- Compiled and provided all textures, prefab static mesh models, and CPF 3D Studio Max models to DRDC Atlantic on CD.

Figure 2: Inside Bridge looking at two Virtual Camera displays

3. Software Issues and Resolutions

Several gaming engines were considered for this project, however, only the Unreal Tournament 2004 editor (UnrealEd) was capable of supporting the requirement for 32 online players.

UnrealEd, the tool used for creating the first person gaming simulation for the Force Protection TD project, has proved to be a highly unstable development environment. UnrealEd's instability has delayed the timeline of the project through constant crashing occasionally leading to constant re-work of the game design. Unfortunately, no official development support is offered for UnrealED unless you have purchased an Unreal Tournament game license and the software does not contain a basic HELP file. The Unreal Tournament game license was not acquired because it was beyond the financial budget of the project (e.g., $450, 000 for a single game license). The majority of the issues (detailed below) were resolved by searching internet gaming and 3D graphic user forums and by general troubleshooting by Human*systems*®'s modeller (sometimes by trial and error).

3.1 VRML model of Halifax CPF

Initially, we believed that importing the provided VRML model of the Halifax CPF would save significant time. Although the VRML frigate (ASE format) was successfully imported into UnrealEd as a 3D static mesh solid, much time was spent trying (unsuccessfully) to determine how to convert the mesh model to CSG brushes for the purpose of modifying and creating interior spaces. The CSG mesh-to-brush conversion was necessary to create the interior spaces of the bridge and hangar (plus associated passage ways) because space could not be subtracted from the solid mesh model once in UnrealEd. Therefore, we attempted to re-build the frigate in CSG brushes to be able to create the interior spaces necessary for the simulation.

We disassembled the frigate into component parts in 3D Studio Max and imported parts that could remain as mesh solids into UnRealEd. Other components (i.e., ship exterior sections) were built as brush models within UnrealEd so that they could be transitioned into interior spaces more quickly. Unfortunately this process was unsuccessful due to a programming defect within the Unreal editor which prevented CSG brushes to share vertices. The frigate created with CSG brushes contained gaps at the vertices. Ultimately, we moved back to 3D Studio Max to re-construct all the frigate's structural sections with the interior space included to avoid the vertices problem. The structural components were then imported back into UnrealEd as textured static meshes.

The sections below describe the design issues in more detail.

1) Issue: The VRML frigate model had to be converted into ASE format to appear inside the Unreal Tournament 2004 world. As a result, the frigate ended up being coloured a flat grey, with some darker areas for the helicopter pad and the slot cover for the front gun. But even with the flat textures, the UnrealEd generated errors and produced odd angular striping patterns on the deck. Further, the Unreal engine did not align the vertices of all the polygons correctly, leaving gaping holes, cracks, and other voids offering visibility to other parts of the ship through walls and floors.

Resolution: Re-create structural sections of frigate using CSG brushes (in the end, this was not a viable solution as detailed later in this document).

Figure 3: Type 1 geometry errors- StrikeForce Mod

2) **Issue**: As noted above, importing the provided VRML model (converted to ASE format) into UnrealEd created major design problems: holes and leaks were displayed in the model; model contained incorrect geometry; no editing capability to add doors, windows and other subtractive geometric modifications; limitations with portals; and the inability to re-texture components of the ship. Also, the polygons imported from the VRML model did not react as expected with the boolean operations normally used to modify the geometry of an existing model. The initial solution was to rebuild the ship with CSG brushes, using the VRML model as a guideline. CSG brushes of complex shapes and angles were required to achieve a reasonably realistic rendering of the frigate. In UnrealEd, these shapes are built so that vertices align pixel for pixel. When compiled for game play, the vertices became misaligned creating gaps and holes in the model. It was discovered that CSG brushes do not share vertices (as previously believed) which resulted in a model full of open gaps and holes. Since no gaps were present in the UnrealEd browser, patches to the gaps had to be investigated and repaired one at a time, requiring additional effort and delays.

Resolution: After many failed attempts to align the CSG brushes, the only solution was to completely modify and build the CPF model in 3D Studio Max.

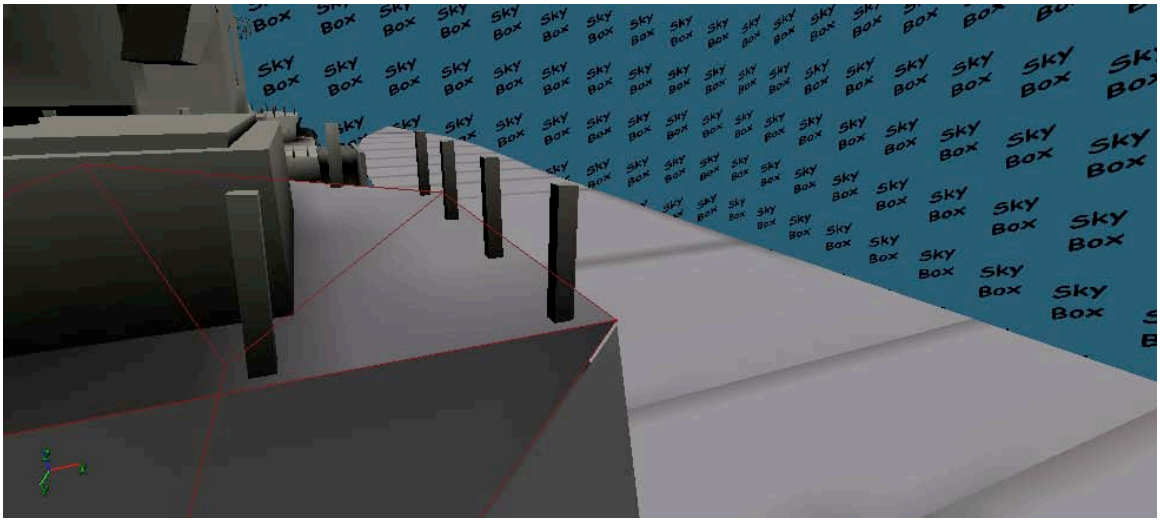


Figure 4: Type 1 geometry error- UnrealEditor

3) **Issue**: The VRML model did not permit any future modification in UnrealEd due to the requirement of importing it as a static mesh. That meant no doors, hatches, windows, or other geometric modifications could be applied to the static mesh in the Unreal editor. As a workaround, the use of zone portals was investigated to facilitate player movement through the solid mesh frigate. The idea behind the use of zone portals was to create an artificial tunnel through a solid barrier. By placing a portal on each side of a static mesh, a virtual door could effectively be created through the mesh. Unfortunately, the zone portals had some critical flaws: 1) some types of weapons and ballistics could not cross through the portal; and 2) in multiplayer mode some players had difficulty crossing the portals producing side-effects such as spinning around 180 degrees, getting stuck on the spot, or bizarre screen flashes disrupting game play.

Resolution: Due to the numerous errors, the use of portals was abandoned. The solution was to re-construct the CFP's structural components including the interior spaces within 3D Studio Max and import the objects back into UnrealEd.

4) **Issue**: The VRML frigate model contained some geometry errors. After closer scrutiny of the CPF schematics, it was determined that important geometric structures, such as the hangar, bridge, and hull, had originally been constructed in VRML with inaccurate dimensions and geometry. This meant that the model we were provided did not match the naval architecture drawings.

Resolution: Parts of the VRML frigate structure were rebuilt including the bridge, hull, and hangar. Theses structures were modified to more accurately reflect the physical dimensions of a real Halifax CPF.

Figure 5: Type 2 error - StrikeForce Mod

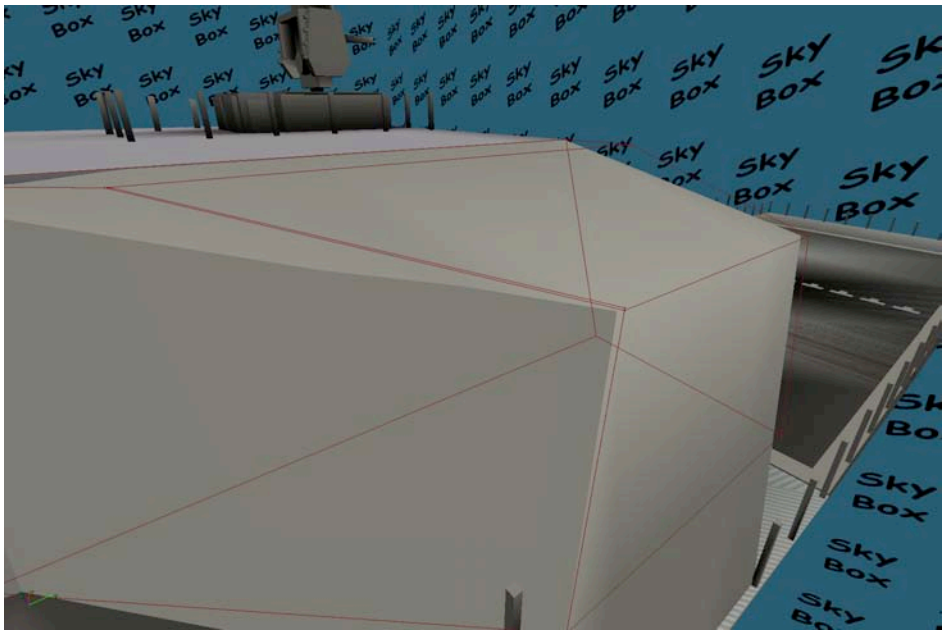


Figure 6: Type 2 error - UnRealEditor

5) **Issue**: In order to correct the geometry of the frigate model, the set of photocopied schematics were referenced. However, the photocopied schematics did not retain the original scaling and provided limited information. The actual scaling did not have an easy conversion, with factors consisting of 1 or 2 decimal places and not following any base 10, base 5, base 2, or base 16

pattern. Each blueprint required a different scale factor because each blueprint photocopy had been completed at a different size.

Resolution: Dimensions of the frigate were completed manually by measuring the blueprints with a ruler and calculating the approximate scale. Acquiring critical dimensions became very time consuming and prone to minor error.

6) **Issue**: The VRML model of the frigate was originally constructed with a large number of polygons. Due to the large number of polygons present, performance in both 3D Studio Max and UnrealEd diminished.

Resolution: A polygon reduction was completed to improve overall game performance.

Figure 7: Ground Error

3.2 BSP errors

When we attempted to reconstruct the frigate within UnrealEd using CSG brushes, the Unreal editor had difficulty managing and interpreting coordinate values when creating complex geometry for the frigate's superstructure. Unreal had difficulty calculating the placement of critical polygon edges and vertices which resulted in Binary Space Partitioning (BSP) errors. BSP is used as a method for resolving polygon order within a world space created by UnrealEd. Polygon order is obtained by cutting the space into convex regions with each cut splitting the world into two sub regions. The BSP process creates a hierarchical data tree structure to store information in relation to non-moveable solid constructive geometry that determines which polygons are visible and which are not. Therefore, in BSP geometry, aligning vertices and edges is extremely important to prevent BSP errors consisting of: leaks, holes, "hall of mirror" effects (texture distortion), ability of player to walk through walls and fall through floors, surface

textures becoming transparent that reveal parts of the map which normally aren't visible, and the creation of invisible undefined spaces that cause players to spontaneously die when entered.

1) Issue: UnrealEd offered only a graphical method of checking alignment when defining brush vertices. As this was the only method available, the rule-of-thumb for defining polygon edges and vertices in UnrealEd was to always snap to a grid of 1:1. Snapping to the grid is the only technique that ensures all coordinate values inside the Unreal editor resolve to integers, eliminating any discrepancy when graphically checking the location of brush vertices.

However, due to the frigate's 3-dimensional angular intersecting geometry, careful thought had to be put into deciding which vertices would be considered "defining corners"(corners with integer X,Y, and Z coordinate values) and which vertices would have "unknown" coordinates that resolve to floating-point values (i.e., number with a decimal point, 10 vs 10.001). Objects could be aligned to a "defining corner", but an "unknown corner" would be for aesthetic use only and could not be used for aligning other objects or intersects.

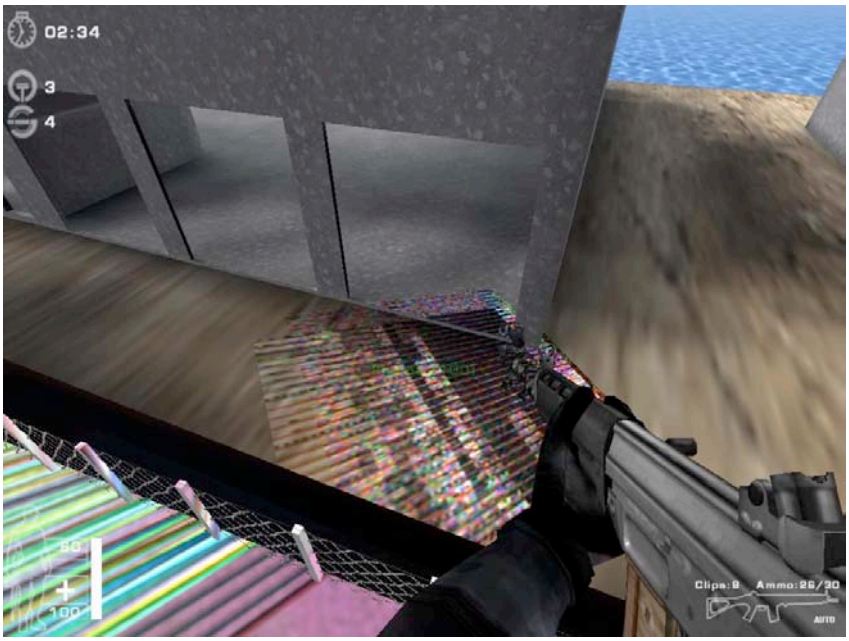


Figure 8: BSP Texture Errors

UnrealEd provides the capability to clip CSG brushes along a plane defined by a pair of vectors. Again, these vectors had to be defined graphically on-screen because there is no feature available for keying-in numeric data. Frequently, the actual plane used to intersect the brushes was NOT coplanar (if they exist on the same geometric plane) with the two vectors. UnrealEd set its own random offset, resulting in an inefficient trial-and-error process of performing some brush clips.

If both defining vectors were not coplanar with the world coordinate axes or collinear with the world coordinate origin, the resulting brushes from the clip operation were left with invalid

geometry causing BSP holes, leaks, invisible Zones of Death, Hall Of Mirrors, or even crashing UnrealEd or the Unreal Tournament 2004 game. The cause for this issue remains unknown.

New methods were investigated in an attempt to redefine which corners would be "defining corners" and which ones were "unknown". Various changes to orders of operations on brush clips and combining brushes clipped on a single axis with a boolean operation against a second subtractive brush proved unsuccessful. A final process was tried by creating brushes by means of vertex editing.

Because CSG brushes can not contain any concave surfaces in BSP geometry, the only method to ensure a complex shape that does not consist of invalid surfaces is to break it down into trihedral, prismatic, and/or bevelled primitives. UnrealEd only provides the capability to create primitives that are polyhedrons, prisms, or polygons extruded as bevels. Breaking down the shape into primitives guarantees the sides to be coplanar (i.e., remain at integer values when created manually with the vertex editing tool).

Within Unreal editor, the geometry constructed with the CSG brushes (by vertex editing) appeared successful, but UnrealEd ultimately failed to compile it properly in the Strike Force game mod. The compiled world was full of BSP holes, leaks, and invisible Zones of Death errors, similar to how the imported VRML frigate was originally rendered. The intersection of planes left gaps, and shapes were not positioned at the specified location. Some of the brushes had misalignments that appeared to be as large as an entire Unreal Unit (translates into 20 mm real-world measurements).

Resolution: Considering all the coincident vertices defining multiple brushes were centred on integer world coordinates, this error appears to be due to problems in the Unreal engine arithmetic - either with the UnrealEd browsers or when the level is complied to play within Unreal Tournament. An attempt was made to copy a functional mirror-image brush (i.e., performing a mirror-image flip) and replacing the defective brush with the newly created copy. However, UnrealEd confused the vertices and constructed another invalid brush (though a uniquely invalid one at that).

These BSP errors are still yet to be resolved - technically the errors should not exist because all the vertices are correctly defined in a valid BSP tree structure. Even the "Check Map for Errors" feature in UnrealEd does not find these BSP errors. They only become clearly evident when the level is compiled for game play.

3.3 Textures

1) Issue: Occasionally, when static mesh models are exported from 3D Studio Max (in ASE format) and imported into UnrealEd, the UV texture mapping coordinates do not function correctly. The texture is properly aligned on the object in 3D Studio Max, but once imported into UnrealEd, the texture pixels become stretched over one dimension. The cause of this error is unknown.

Resolution: The only resolution that proves 100% successful is to rebuild the model, recreate the UV mapping coordinates, and import the static mesh again.

2) Issue: UnrealEd has trouble importing texture properties of static mesh objects from 3D Studio Max. When UnrealEd imports an ASE file, texture information (e.g., cropping, scaling, alignment, etc.) is lost. As a result, static meshes are displayed with a default UnrealEd texture. The only aesthetic modification that can be performed in UnrealEd is to replace all the surfaces with a uniform texture file.

Resolution: Due to 3D Studio Max's ASE export limitation, no shading properties (i.e., lighting and glossy properties) were assigned to any textures or static mesh objects. Static meshes and associated textures were worked on separately – static meshes were constructed and modified within 3D Studio Max and textures were developed using an external paint program. Texture alignments (applied to static mesh) were completed in conjunction with the creation of the texture itself.

3) Issue: Regardless of the type and shape of brush, there is an issue with the vertex lighting. Although lighting is not critical to the operation of the 3D environment, the vertex lighting calculations appeared to be incorrect for all the brush surfaces. A brush with a uniform texture is rendered in UnrealEd with visible dark bands (striping) around the edges of the faces, even when the face is coplanar with an adjacent brush with coincident vertices. This issue also applies to static meshes.

Resolution: The lighting problem is still unresolved.

3.4 Static Meshes

1) Issue: All objects created within UnrealEd are created as CSG (constructive solid geometry) brushes. Although UnrealEd has a feature to convert objects between static mesh and CSG brush format, the mesh-to-brush conversion did not work and the brush-to-mesh conversion was full of bugs and very unreliable, causing the editor to crash most of the time.

Resolution: It was determined that brushes and meshes had to be essentially considered as two totally separate types of non-commutable entities. So, using UnrealEd's feature to create complex static meshes from CSG brushes (e.g., reconstructing the hangar in CSG brushes to facilitate the addition of interior spaces at a later date) was not a viable option. Therefore, geometrically complex objects were modelled in 3D Studio Max and were imported into UnrealEd as a static mesh.

2) Issue: Static meshes can not be edited once they are imported into UnrealEd. UnrealEd has the capability to rotate, rescale, and apply a uniform texture over the entire object, but it does not permit any modifications to the pre-existing geometry. Static meshes can coincide with CSG brushes, however, meshes are always considered additive, and subtractive CSG brushes can not subtract matter from a mesh. Essentially meshes and brushes have no interaction between them.

Resolution: Any modifications to the geometry of a static mesh were completed within 3D Studio Max. Once complete, the static mesh objects were imported back into UnrealEd in the ASE format.

3.5 Micellaneous

3.5.1 Player Bounding Box

1) Issue: The bounding box encapsulating the space around the player in Unreal is too large to allow players to traverse some doorways and interior rooms (specifically the bridge) on the frigate.

Resolution: Attempts were made to modify the existing bounding box (i.e., reduce the width) of the player to accommodate narrow doorways and tight interior spaces. The class names of the bounding box variables were identified, but we were unable to locate the bounding box parameters within the game code to manipulate the values. As an alternative solution, narrow doorways were increased slightly in width to accommodate the player bounding box. Also, components of the bridge were modified to appear solid but allow players to pass through them. We were concerned that this may have created unrealistic sightlines for defensive planning purposes, but it has had a minimal impact.

3.5.2 Breaking Glass

1) Issue: The glass added to the bridge windows was not shattering when broken.

Resolution: An attempt was made to modify the glass emitters (creates appearance of glass shards when broken). Unfortunately, the glass breaking properties did not reflect the changes successfully and the windows just disappear when broken.

3.5.3 Security Camera and Monitors

1) Issue: Video captured by the security cameras did not transmit to all the network players. Only one person on the game network could see the captured video on the security monitor.

Resolution: The camera emitter program code was modified to facilitate video transmission from the security cameras so all network players could see the video on the security monitors.

2) Issue: You can not see your own player on a security monitor.

Resolution: No solution has been discovered to correct this problem. However, the issue has a relatively small negative impact on the effectiveness of the security cameras since we are unlikely to place the security monitor in the view of the security camera it is displaying.

3.5.4 Game Packages

1) Issue: When a local level is loaded into UnrealEd, the editor automatically only loads the objects that are currently being used in the level. However, objects that exist in the game package can still be accessed and modified when developing the level further. Unfortunately, changes made to an object are lost if it was not initially loaded in the Unreal browsers.

Resolution: An order of operations was determined to workaround this problem. The order of operations is as follows: 1) load the level, 2) open the textures package, and 3) open the static mesh packages. This order of operations forced the UnrealEd browser to read all files in the game package regardless if the objects are used in the level or not. Therefore, if an object is added to a local level and then modified, the changes remain permanent.

2) Issue: UnrealEd spontaneously (and frequently) crashes resulting in toolbar controls (e.g., "Snap to grid") no longer functioning, the creation of error messages (e.g., critical errors, runtime errors, illegal operations), and the failure to import static meshes (for objects created in programs outside of UnrealEd).

Resolution: These are unknown stability issues and there is still no logical explanation. The only viable solution is to reboot UnrealEd.

3) Issue: Often an object would appear in the Unreal editor, but not appear in the game despite saving and reviewing all the object references. Also, after an object successfully appeared in the game, it disappeared the next time after an unrelated aspect of the world was modified and re-saved.

Resolution: In order to reduce the occurrence of this issue, a specific mode of operation was developed – 1) Objects put in Unreal browser, 2) render objects with UnrealED, 3) rendering objects forces UnrealEd to read from the package file only, and 4) Objects are saved into game package.

3.5.5 UnrealEd Interface

1) Issue: Occasionally, no view ports (multiple design windows) would appear during startup of UnrealEd. As a result, no workspaces were displayed within the Unreal Editor to produce work.

Resolution: Changing the view port settings (through the VIEWPORT menu) sometimes prompted UnrealEd to rebuild the interface and display the design windows. Unfortunately, this method was not 100 percent reliable. If changing the view port settings failed, the UnrealEd application was restarted.

2) Issue: Too many consecutive *undo* commands UnrealEd to crash.

Resolution: Tried to avoid making consecutive *undo* commands.

3) Issue: Unfortunately UnrealEd does not indicate any numeric value on the grids and there is no position indicator for the cursor. While the X, Y, and Z coordinates of the origin of a brush and static mesh can be checked, the actual coordinates of the vertices can not.

Resolution: The only way to obtain coordinate data was to lay a ruler across the front of the computer monitor and count the grid lines. At a later date, a ruler feature was discovered within UnrealEd that provided a graphical representation of a ruler to be manipulated on screen. This method slowed down work progress because points had to be checked visually on the screen.

4. Recommended Way Forward

A considerable amount of development time was exhausted trying to successfully construct the CPF model within the Unreal game editor. As stated previously, it became necessary to reconstruct the CPF model in 3D Studio Max to mitigate UnrealEd's limitations and programming errors when handling complex object geometry. Reconstructing the frigate took time, but it has produced some significant advantages:

- The CPF VRML model was reformatted into a standardized 3D gaming geometry. The CPF model now has the ability to be ported to other game engines or 3D graphic design applications. Use is not limited to only Unreal Tournament 2004.
- Realistic textures were created from the CPF photographs and stored in the TARGA format. The TARGA format allows a variety of image editing applications access to the texture for modification and use in multiple game engines.
- The 3D Studio Max model of the CPF and its related components have proper texturing compared to the VRML model which contains no textures.
- Knowledge gained through this project has generated a core capability that can now be efficiently applied to any further modifications and developments to the CPF model.

Based on the existing CPF model and capabilities developed during this project we would recommend the following development options as possible ways forward.

4.1 CPF Model Enhancement

The 3-Dimensional Halifax class Canadian Patrol Frigate model could be further developed in a 3D modelling environment such as 3D Studio Max or equivalent application. Model enhancements could include:

- Creation of additional textures for exterior sections of ship (e.g., missile tubes, vents, hatches, gun turrets, flight deck, rope, piping, etc.).
- Creation of additional textures for interior sections of ship (e.g., passage ways, rooms, etc.).
- Construction of additional interior sections of frigate.
- Addition of 3D static mesh objects (e.g., valves, chairs, pipes, lights, etc.) instead of using expansive, flat texture maps.

4.2 Dockyard Enhancement

Budget constraints limited the development of the existing dockyard, and as a result, the surrounding environment could be further enhanced to increase the realism of the simulation. A variety of prefab static meshes and textures are available within the Strike Force modification to improve the visual representation of the port. Dockyard enhancements could include:

- Addition of a cityscape or city skyline to enhance the distant view perspective and realism of the skybox.
- Addition of midrange city buildings to provide realistic three-dimensional content and depth to the mid-ground and to provide additional manoeuvre space for ground forces.
- Addition of street lights, ISO shipping containers, barrels, signs, cranes, and dockyard specific vehicles.
- Develop AI bots to populate the dockyard with civilians and labourers.

4.3 Game Engine

Unreal Tournament 2004 was selected for the project because it was the only game engine that explicitly stated it could support 32 networked players. Given UnrealEd's volatility and lack of technical support, it may be beneficial to explore other 3D commercial game engines that provide more stability and offer a greater support network. If this is not an option, an upgrade for UnrealEd should be investigated (at the time of this writing, no new upgrades were available). Also, Unreal Tournament 2004's real-time voice chat and 32 multiplayer features should be properly tested.

4.4 Scenario Modification

As a tactical simulator, Strike Force is a good starting point for creating game characters with appropriate weapons and uniforms. The Strike Force game modification requires some customization to accurately reflect the Canadian Forces. General modifications include: skinning game characters to reflect Canadian Navy uniforms, creating Canadian issue weapons, and changing the game start-up splash screens and menus.

Other tactical scenarios could also be developed to investigate other CPF force protection situations. For example, scenarios could be developed to simulate a water-borne assault while at anchor in a harbour or a heli-borne assault at sea, and so forth.

4.5 Force Protection Evaluations

Ultimately, the CPF simulator could be employed for the benefit of developing standard operating procedures (SOPs) for force protection and then training those SOPs to ship's crew in a simulation environment. The simulator could also be used to select, develop, and integrate different technologies to support the phases of a force protection engagement: detection, surveillance, and repelling a threat.

> Detection: The current CPF model and gaming environment can be programmed to simulate a variety of early-warning perimeter defence technologies that can be used to alarm and inform the force protection crew of a security breach.
>
> Surveillance: As the current model demonstrates, video camera surveillance can be incorporated into the CPF model to provide multiple remote sensing capabilities to a single force protection crew member. Surveillance capabilities could be expanded to include auditory sensing and other visual spectra (e.g. night vision, thermal imagery).

Repelling a Threat: Within the current CPF model, a crew member in the bridge can observe multiple video surveillance cameras including a roaming UAV camera. However, having determined that security has been breached and observing an enemy force, the crew member must currently deploy force protection based solely on SOPs. Various decision aids could be incorporated to support these force deployment decisions.

The selection, development, and programming of these different technologies into the CPF model can be readily achieved through relatively inexpensive programming but the determination of the utility and usability of these capabilities, in the context of ship's operations, requires valid, reliable experimentation with representative users (i.e. sailors), performing realistic tasks in a realistic simulation environment. For example, we have demonstrated that video surveillance can be provided and displayed in the model but how can we optimize the benefits of this technology for force protection?; what number of cameras achieves the best results?; what are the best locations for the cameras?; how might a UAV be employed?; is the bridge the best location for monitoring the cameras and coordinating the deployment of protection forces?; can the monitoring task be incorporated into an existing tasking on the bridge or is a dedicated sailor required?; and so on.

There are many questions that could be investigated using the model with sailors integral to the experimentation process but quality experimentation requires the collection and analysis of objective and subjective data. Ideally, measures of effectiveness and measures of performance would be derived that could be collected and collated in real time by the gaming model. This would require the development and programming of a data capture suite of tools to support these experiments.

The most significant benefits of employing commercial gaming software to simulate force protection issues in a CFP model are the low cost platforms, low cost software, inexpensive programming, short development timeframes, high flexibility and functionality, and high quality realism and imagery. The downside is the extra development time and effort required to overcome software deficiencies and to develop unique work-around solutions. However, once this initial investment has been made subsequent changes and developments can reap the benefits.

References

[1] Frim.J, and Thompson, M.H. Assessment of 1st Person Shooter Game Engines for Force Protection Applications. DRDC Atlantic CR 2005-088, January 2005.

[2] Unreal Engine 2 _Unreal Technology_ [Online] Available http://www.unrealtechnology.com/flash/technology/ue2.shtml, Jan 2, 2005.

List of symbols/abbreviations/acronyms/initialisms

BPS	Binary Space Partitioning
CPF	Canadian Patrol Frigate
COTS	Commercial off the shelf
DND	Department of National Defence
FPS	First Person Shooter
OPI	Office of Primary Interest
R&D	Research & Development
UAV	Unmanned Air Vehicle
VRML	Virtual reality markup language

Distribution list

Document No.: DRDC Atlantic CR 2005-197

LIST PART 1: Internal Distribution by Centre:

1 Mark Hazen
1 Allan Gillis
6 DRDC Atlantic Library
2 HumanSystems Inc.

10 TOTAL LIST PART 1

LIST PART 2: External Distribution by DRDKIM

1 DRDKIM

1 TOTAL LIST PART 2

11 TOTAL COPIES REQUIRED

This page intentionally left blank.

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATOR (The name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.)

Human Systems Inc.
111 Farquhar St., 2nd Floor
Guelph, ON N1H 2N4

2. SECURITY CLASSIFICATION (Overall security classification of the document including special warning terms if applicable.)

Unclassified

3. TITLE (The complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C, R or U) in parentheses after the title.)

Development of a CPF Model in UnReal Tournament

4. AUTHORS (last name, followed by initials – ranks, titles, etc. not to be used)

Bennet, J., Frim, J. and Tack, D.

5. DATE OF PUBLICATION (Month and year of publication of document.)	6a. NO. OF PAGES (Total containing information, including Annexes, Appendices, etc.)	6b. NO. OF REFS (Total cited in document.)
January 2006	17	2

7. DESCRIPTIVE NOTES (The category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Contract Report

8. SPONSORING ACTIVITY (The name of the department project office or laboratory sponsoring the research and development – include address.)

DRDC Atlantic, 9 Grove St Dartmouth NS

9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.)	9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.)
11BK	W7707-4-2677A

10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.)	10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.)
	DRDC Atlantic CR 2005-197

11. DOCUMENT AVAILABILITY (Any limitations on further dissemination of the document, other than those imposed by security classification.)

(X) Unlimited distribution
( ) Defence departments and defence contractors; further distribution only as approved
( ) Defence departments and Canadian defence contractors; further distribution only as approved
( ) Government departments and agencies; further distribution only as approved
( ) Defence departments; further distribution only as approved
( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11) is possible, a wider announcement audience may be selected.))

13. ABSTRACT (A brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual.)

A 3D model of a Canadian Patrol Frigate (CPF) was imported into a version of the UnReal Tournament multi-player game. This document gives details of the problems encountered and resolutions developed during the import process. The result of the project is a set of playable levels that represent the upper deck, hanger and bridge of a Halifax class frigate.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

First Person Shooter Game
CPF Model
UnReal

This page intentionally left blank.

Defence R&D Canada

Canada's leader in defence
and National Security
Science and Technology

R & D pour la défense Canada

Chef de file au Canada en matière
de science et de technologie pour
la défense et la sécurité nationale

www.drdc-rddc.gc.ca