Space Marines

A Distributed Online Shooter

Space Marines is a two-dimensional shooter whose aim is to provide a healthy conduit to the pursuit of happiness for all ages. It is implemented using a distributed peer-to-peer architecture and seeks to abate the role of latency by utilizing concepts found in dead reckoning to extrapolate entity positioning.

Space Marines

A Distributed Online Shooter

# Introduction

Space Marines is a two-dimensional shooter in which the primary objective is to blow other players up.  The application is written in C# and leverages the Microsoft Silverlight application framework to enable the game to be played easily online.  The underlying network architecture utilizes a simple server based system to set up games and provides any necessary files to the clients but primarily communicates between clients in a peer-to-peer model utilizing sockets officiated by the transmission control protocol (TCP).  The application will be able to handle moving and shooting in real time and will resolve issues with latency between clients using the Dead Reckoning algorithm in order to minimize network traffic.  As the project expands additional complimentary game features will be explored including novel weapons, upgradable ships, and additional gameplay styles.    
  
Problem  
Distributed systems can be defined as a system “in which hardware or software components located at networked computers communicate and coordinate their actions only by passing messages” (Dollimore, Kindberg, & Coulouris, 2011).  Distributed systems bring with them a host of boons including the ability to share resources remotely, allow clients to dispense with local storage and computational needs as they acquire these by perhaps renting services from a cloud, and even allow more seamless and collaborative communication between teams of people located in different continents.  With the advent and continued proliferation of such systems, it is of greatest interest that distributed systems architect remain cognizant of their challenges.  These challenges include not only the fact that communication between processes in the system through the channels of the perhaps the internet suffer from high latency, reliability concerns, and the malicious attacks, but the heterogeneity, scalability, extendibility, security, scalability, partial failure, concurrency, and transparencies to the end user present major design issues as well (Dollimore, Kindberg, & Coulouris, 2011).     
  
Not only do popular applications domains such as finance and commerce, informational society, education, transport and logistics, and the sciences maintain the need for distributed systems but multiplayer gaming also hinges on the need for distributed systems and complex distributed algorithms and hardware configurations to support player populations in upwards of tens of thousands.  Many of the distinct distributed challenges for gaming include the “the need for fast response times to preserve the user experience of the game, real time propagation of events to many players, and maintaining a consistent view of the shared world” (Dollimore, Kindberg, & Coulouris, 2011).   
  
The desire to create Space Marines stems uniquely from these challenges.  It is desirable to not only investigate how an online game presents a unique challenge to distributed systems in general, but how the unique need for true real time systems should be handled.  One of the most basic problems in a distributed system is no shared memory between clients, and this case will be no exception.  Therefore, it is essential in the presented distributed game to establish a method of passing messages that will allow the clients to communicate with one another.  Space Marines will solve this using one of the lowest level solutions - using sockets and the TCP protocol.  To achieve the best possible performance, all messages will be sent as infrequently as possible, and carry a minimal amount of data.  
  
The choice of building a game was enticing due to the fact that it would not just be entertaining to develop and play but would test and refine programming ability.  However, a constant in almost every multiplayer game is latency.  Most distributed games approach latency resolution in a large variety of ways but true real time gaming is a difficult concept to implement properly.  Major points of contention include handling concurrency issues as players in real time concurrently modify the game world, preventing congestion due to the rapid rate of updates, and keeping the global game state properly propagated and updated.  Space Marines attempts to solve each of these issues by utilizing peer-to-peer methodology and enforcing each peer to keep his/her entities updated on the other peers’ systems by sending period updates to these cloned entities should projected positions be too far from the actual positions.  This is known as *Dead Reckoning* (Aronson, 1997) and will be discussed in the following sections.  
  
Space Marines  
Gameplay Mechanics  
Space Marines is a dynamic real time multiplayer game.  As users navigate to the website, a user will be immediately ushered into the heated gameplay currently taking place.  Upon entering, the player will be given a unique color that is differentiable enough from the other colors in the system and appear as a small circle painted with this color on the game playing field.  The game playing field which itself is simply a static sized rectangle similarly sized for all remote players.  Players will move their own character in game by using the common set of WASD keys made popular by FPS games.  Holding W, A, S, or D will allow the user to move north, west, south, and east along the playing field, respectively.  It is important to note that when none of these keys are pressed the player is immediately made immobile--there will be no velocity compounded the longer a user holds these keys.  In order to aim, the user simply hovers his/her mouse pointer over a location on the game field and clicks the left mouse button.  Holding this button will release a steady stream of bullets in mouse pointer’s direction.  Bullets’ position over time will be dictated by a pre-established bullet velocity for that particular gun and a direction determined by the current player and the mouse pointer positions.  Bullets themselves will be color coded with the same color as the player.  The goal of the game is a simple one--destroy all enemy players.  Thus, each player character will have a pre-determined value of health and each bullet will take off a percentage of health from the enemy.  As players reach 0 health points, he/she is removed from the battle until the next round of gameplay.      
  
Enhancements  
A variety of enhancements will be added to the game when the base gameplay has been completed and rigorously tested.  These preliminary enhancements to the game will include the following if time permits:  During the course of a round, small squares will be dropped on a random position in the game board but no closer than a preset distance to any character.  The color of the box will denote the type of upgrade to be obtained by the player who is successful in capturing the device.  One color will denote a health upgrade, while another color might denote stamina upgrades to temporarily enhance speed and damage.  To capture the upgrade players must damage the box enough to reveal its contents, which will appear as a small round capsule.  Players must then directly move their characters over the upgrade to gain its benefits; once a capsule is consumed by exactly one player, the capsule disappears from the game board.  To give proper motivation for eliminating other players besides the obvious goal of winning, weapon upgrades will be given to players as they defeat enemy players.  However, if the player dies, all weapon upgrades are reset and the player must again work up his prowess on the battlefield.  Other additions to the game will perhaps include placing small immobile and invincible level objects on the playing field that will shoot random bursts of bullets in patterns across the map for human players to avoid.  Finally other game types in addition to the traditional slayer mode may be investigated as time permits.    
  
Methods  
The individual mechanics for our game involve the overarching use of dead reckoning.  Thus it naturally follows that the game will also support a peer-to-peer component as well and in fact it does.  In keeping with this scheme, the server’s role was minimized as much as possible while individual peers manage themselves collectively as a group to keep a copy of the game state synchronized and up to date on peer computers while mitigating the potential devastating effects of low latency of the internet, or the main communication medium by which the game interacts, as much as possible.  The term devastating effect is used primarily because in a true real time environment any observance of lag or “stuttering” degrades the gameplay experience and disrupts professionalism of the service.    
  
As players enter the game, they inform the server of their arrival.  The server responds by firstly updating its list of IP addresses to include the newest player.  It then sends the full list to the newest arriving peer, while sending the new player’s IP address to all other peers in the system.  From this point the server remains on standby with minimal interaction with the peers in the peer-to-peer system.  Notice that as a tracker is the only “server” concept in the peer-to-peer bit torrent application, so is the list of IP addresses in this system.  The newest player, let us call him peer X, is now placed in the middle of battle.  Peer X creates his own character entity along with a virtual copy of that entity and sends the virtual copy to all other peers to be included in their individual entities.  All other peers share with peer X their owning entities upon reception of the new IP address from the server.  Entities consist of a position, direction, and a movement function--this is usually a straight line and may be any potentially dynamic game object.    
  
Peer X now owns his character entity, namely C.  Assume now that X sends a steady stream of bullets out of C to opposing players.  Each bullet, B, created by X, will be created and propagated throughout the system to all peers in the same manner as described for the entity C.  Note that only peer X contains actual instances of C and B (along with a virtual copy) while all other peers have only the virtual copy.  All peers including peer X extrapolate virtual entities positions by means of the movement function given.  Should the actual position of C or B deviate too far from the predetermined threshold, X will send a direct update of C or B to all peers.  Collision detection is done by peers of the owned entities involved in the collision.    
  
By these mechanisms as dictated by Dead Reckoning, all peers collectively donate their resources to the game’s functioning and further take part in maintaining the global game state.  The injurious effects of high latency are mitigated by depending on each peer to extrapolate or predict other peers’ actions.  Unprecedented quality and seamless responsiveness is promulgated in the thrilling game ubiquitously known as Space Marines: The Final Frontier.

Milestones  
*Each task will be completed by the date listed on the same line.*

* 9/16/2011: Setup the initial project structure with client and server and upload to googlecode.
* 9/23/2011: Implement basic structure of message passing by sockets and implement IP address event pushes from server to clients whenever a new player joins the game.
* 9/30/2011: Allow players to enter the game field and move freely around on game board. Rigorously test and refine to ensure smoothness of player movement by adding smoothing logic to dead reckoning algorithm.
* 10/7/2011: Implement targeting and firing of bullets.
* 10/14/2011: Implement collision detection with health removal.
* 10/21/2011: Implement player death and round resetting.
* 10/28/2011: Implement player health indicators (a number or a progress bar above each character).
* 11/4/2011: Implement weapon upgrade logic.
* 11/11/2011: Implement health gain logic by capturing the health capsules.
* 11/18/2011: Level logic with enemy bullets.
* 11/25/2011: Detailed outline of final paper with project member assignments.
* 12/2/2011: Final paper completed.
* 12/12/2011: **Enhancements (Implemented as time allows)**
  + Stamina buff logic (confers temporary speed and weapon damage gains).
  + Investigate additional game types.

Bibliography

Anderson, C. (2010). *Pro Business Applications with Silverlight 4.* Apress.

Aronson, J. (1997, September 19). *Dead Reckoning: Latency Hiding for Networked Games*. Retrieved September 14, 2011, from Gamasutra: http://www.gamasutra.com/view/feature/3230/dead\_reckoning\_latency\_hiding\_for\_.php

Dollimore, J., Kindberg, T., & Coulouris, G. (2011). *Distributed Systems: Concepts and Design.* Addison Wesley.

MacDonald, M. (2010). *Pro Silverlight 4 in C#.* Apress.