LCO Milestone Review: Antenna Ball Mayhem

Christian Bell, Thor Carpenter, Peter-Michael Osera
{cj, thorc, psosera}@cs.washington.edu

January 11, 2005

1 Abstract

Antenna Ball Mayhem (working title) is a concept for a web-based, video game in which multiple antenna balls, all player controlled and affixed to a car driven by a crazy driver, attempt to dodge and knock each other into oncoming debris in last-man standing type competition. From a player standpoint, the game offers simple, yet addictive gameplay that stands out from it's web-based peers in it's direct competition model of gameplay. From a developer standpoint, the game builds outward from a simple gameplay core that can be easily extended to approach the quality of a AAA-quality game (e.g., graphics, music, etc.), but does not require them to be fully realized. And from an investor standpoint, such a game built on the web-based model naturally takes advantage of the advertisement-based revenue commonly found in web-based games. This combination of value makes *Antenna Ball Mayhem* a worthwhile endeavour.

2 Operational Concept

Our game concept draws upon the pre-existing user community of web-based video games in the same vein as the games produced and hosted by companies such as PopCap Games, MSN Games, and Yahoo! Games. The average user of such a game is a "casual gamer", one who plays easily accessible video games typically rated "E for everyone", in short bursts. Such a user plays these games from an Internet-capable computer, not just at home, but also at work and other technology-enabled areas when they feel the urge to do so.

Our concept, *Antenna Ball Mayhem*, draws on these features of the user community to make a compelling experience for the player. In this game, the player controls one of many antenna balls (of an arbitrary amount from tens to perhaps even hundreds of player-controlled balls if feasible) affixed to the hood of a moving car by antennae from a third-person perspective (rendered in a 2.5-d style, scalable to full 3d if time allows) of his or her ball and the road ahead. The

driver of the car is a crazy individual who entertains himself by attaching his collection of antenna balls to his car and driving around with reckless abandon in an attempt to destroy said balls with flying debris, wreckage, and anything other crazy things he finds in his travels. The player must preserve his or her antenna ball by moving it out of the way of the flying objects, knocking his or her peers into or away from oncoming debris.

This simple concept will appeal to the player with it's combination of simplicity, minimal system requirements (as is the norm with web-based video games), and multiplayer competitive-cooperative gameplay previously untouched in this sector of video games. And since this game is based on the established model of web-based video games, investors can take time-tested approaches to this game such as revenue-by-advertisement, whether that be within the game (flying Coke cans) or outside of the game (traditional ad banners).

3 System Requirements

The essential *Antenna Ball Mayhem* concept supports the gameplay described above which requires some sort of network play, either over a LAN or the Internet and basic 2D graphics and sounds. From the publisher/developer perspective, the game will require online distribution (i.e., bandwidth costs, staffing, support), either in the form of a simple client-server download in which the user community is responsible for creation and hosting of game sessions (ala Quake, Counter-Strike) or a more complex hosting service where the game is not only distributed online, but a matchmaking service and game sessions are also hosted online by the publisher (ala Blizzard's Battle.Net, Halo 2's matchmaking service on XBox Live). From the client end, the nature of the game and the user community implies that we target a minimum of computer requirements, so as a rough estimation, we expect the client to have an Internet-enabled budget computer from the last five years. Ideally, we could expand the game concept to encompass multiple platforms (e.g., cell phones), but as a conservative estimation, Internet-enabled budget computers are a minimum. If play testing proves it feasible we would like to support users with at least a 56kbps dial-up connection.

In terms of reliability requirements, the player expects a seamless game experience, i.e., no crashing, responsive gameplay, consistent framerates, and little-to-no "lag" given an adequate Internet connection. In addition to these basic constraints on the game itself, the player will also expect developer maintenance in the form of patches to fix gameplay issues and bugs. If a matchmaking/game hosting service is undertaken, then the player will also expect that the service be open and available all the time, barring maintenance-related downtime and acts of God. Conversely, from the service-provider's perspective, the matchmaking system would need to be reliable insofar as it requires as little maintenence as possible.

From this, we define a set of priorities that should be tackled in order when realizing this game concept. First, the basic gameplay should be realized. This involves the construction of a client and server with a simple game engine involving basic graphics, sound, and network play. In this manner, we can, throughout the rest of development, ensure that the experience from the client is as seamless and fun as possible. This will require performance analysis and gameplay testing to ensure that the basic client and server meet this criteria. Once the client and server are deemed to meet this criteria, then development can proceed to other ventures. If a matchmaking service is undertaken, then that infrastructure is of next highest priority, which also needs to provide seamless gameplay to the end-user. Finally, advanced graphics and sound assets can be added as necessary to raise the quality of the game.

4 System and Software Architecture

We define one flexible approach to the implementation of this concept and its rationale. We divide the system into two major components: the client and the server. If we utilize a matchmaking service, then we introduce a third major component, the infrastructure encompassing the matchmaking service. Since one of the principal aims of this game is to be runnable everywhere, we employ java as our target language for it's native support across a wide variety of platforms and it's support for web-based distribution via JNLP and WebStart. For ease of compatability, we also employ java to write the server software. The matchmaking service can be implemented as a web-based interface with php powered by a java servlet.

Roughly speaking, the server will be in charge of syncing all the clients (players), and the handling of non-client details such maintaining the world environment (spawning of objects, for example). The client will be in charge of graphic and sound rendering and the transmission of the user's input to the server. Also, the server would like to off-load as much work onto the client as possible, such as the handling of physics calculations and position, keeping a balance between security and performance. The matchmaking service would keep tabs on known servers to allow clients to query the service for available game sessions. The result is a triangle-type interaction between the three major components of the overall system.

In addition to these major components, game assets, graphics and sound, need to be created. Off-the-shelf products can be used to create these assets in parts (e.g., image programs, sequencers) but minimally, an in-house level editor program will be required to unite these assets into a cohesive level. Also of interest is the need for small benchmarking and monitoring assets such as logging programs to allow for accurate analysis of the game by developers and support staff.

5 Life-cycle Plan

There are two major classes of stakeholders for this concept. The first is the "casual gamer" consumer, as described in the operational concept, that likes simple, easy-to-access games, but is looking for a direct, multiplayer aspect as well. The second is the investor, which we break further into two classes. The first are the game publishers and content-providers of this genre of web-based video games such as Popcap, MSN Games, and Yahoo! Games. These companies are constantly looking for quality games to add to their libraries. The second class of investors are advertising agents. This game is unique in that it has a real-world aspect to it that allows for natural product placement such as the flying debris including Coke cans, Wrigley gum wrappers, Toyota cars, and so forth. This is in addition to the traditional banner-based advertisement found on the mentioned content-providers' websites.

To concretely describe a life-cycle plan for this concept, we break down the plan into five principal questions:

- *Why is this system being developed?* To create a simple, addictive multiplayer game, targeted towards the casual gamer that is already part of the established user community of web-based video games and the established investors of this community - web-based game content providers and advertisers.
- *What will be done by when?* We prescribe the following plan to execute the development of the game assuming a six week period from initial team organization to final release:
  - Week 1: Detailed planning of system architecture - how do we break the system down into manageable components; define protocols, interfaces, abstractions
  - End of Week 1: Milestone 1 - rough, malleable, "complete" plan of system design
  - Week 2: Assignment of components to smaller teams; design and implementation of those components
  - Week 3: Continued design and implementation of components coupled with integration of components as time and need arise; light development of content assets (graphics, sounds).
  - End of Week 2: Milestone 2 - competition of individual components
  - Week 4: Complete integration of components into a complete system
  - End of Week 4: Milestone 3 - beta release
  - Week 5: Evaluation of the beta release; bug fixing and heavy content development
  - Week 6: Continuation of bug squashing and content development with heavy gameplay testing

– End of Week 6: Mile 4 - final release

In this plan, we spiral three times, once for planning, once for code development, and once more for content development.

- *Who is responsible for a function? Where are they organizationally located?* With a projected group size of eight to ten people, we expect to assign two to three people on each component (such as the graphics engine, level builder, etc.). With differing schedules, we except that much of the group will work independently at home or in their small groups in the labs. In light of this, we will require that the code and assets be located on a mutually accessible cvs server.

- *How will the job be done, technically and managerially?* Technically, we will employ the java programming language with other common tools as agreed upon the group. Because of the scheduling problem, we also use a wiki and instant chat programs for organization purposes. Managerially, we follow the spiral model of development, with the spirals outlined in the six week plan.

- *How much of each resource is necessary?* As described above, we expect to assign two to three people on each major component, working at an estimated two to three hours per day per week.

6 Feasibility Rationale

We identify two major risks with this proposal. The first is common with any game: the concept may not fly in the marketplace. Even great games have a possibility to fall to obscurity, a risk highlighted by the fact that our concept relies on exclusive multiplayer gameplay to prosper. The second concerns the project environment itself. The small time frame in which to execute the project coupled with a group of people that most likely will not have the requisite experience in the areas of graphics and networking is an object of concern.

However, as we have asserted above, the game concept is scalable to many levels of complexity. The required gameplay mechanics and logistics are simple enough to be executed in the given short period of time. This reduces the second risk in twofold: the time required to implement the project is small and consequentially the time invest is also small. The problem of unexperienced programmers is mitigated somewhat by using Java which everyone is familiar with. The first risk is unavoidable, but we feel that we have a compelling, original game that people will enjoy. Furthermore, we have justified above that there is a definite market whose value is worth the aforementioned risk. In this manner, we believe that *Antenna Ball Mayhem* is a worthwhile project to invest in.