Using Recommendation Systems to Adapt Gameplay

Ben Medler, Georgia Institute of Technology

September 15, 2008.

Contact:
Ben Medler
benmedler@gatech.edu

ABSTRACT

Recommendation systems are key components in many web applications (Amazon, Netflix, eHarmony). Each system gathers user input and searches for patterns that exist in order to determine user preferences and tastes. These preferences are then used to recommend other content that a user may enjoy. Games on the other hand are often designed with a one-size-fits-all approach not taking player preferences into account. This paper examines how current web application recommendation systems compare to current games that adapt their gameplay to specific users. The results from this comparison show that games have not to date used certain types of recommendation approaches. Design suggestions for how game developers may exploit the utility of these recommendation features are discussed and examined.

INTRODUCTION

Games are often designed with a one-size-fits-all approach. Developers attempt to design games that will reach as many players in their target audience as possible but the demographics that make up these markets are varied (Herrmann, 2007). Additionally as the amount of online players grows (Herrmann, 2007) developers will need ways to precisely target their current player markets along with new potential markets becoming available. Developers could create niche games that focus on specific groups of players or instead leverage features that can adapt games to specific players in a larger audience. Players who are familiar with personalized content delivered online will come to expect games that mold, or adapt, to their preferences and tastes.

Consequently, recommendation systems filter user input information in order to provide users with suitable content based on their preferences and tastes (Chen et al., 2007). This is accomplished by gathering data, filtering that data, and outputting filtered information based on the given initial inputs. For example a movie recommendation system may gather data about user demographics, user movie taste's, and information about the movies themselves. The data is then correlated and cross checked using different filtering methods in order to search for patterns between the users and movies (Segaran, 2007; Adomavicius. & Tuzhilin, 2005; Vozalis & Margaritis, 2003). The final output to each user is recommendation information indicating other movies that matched the inputs with the search criteria.

Producing output based on user preferences is a common characteristic recommendation systems share with adaptive systems. Adaption occurs when a game delivers output that changes the player's character, non-player characters or the game's environment/statistics based on player information the game gathers (Bailey & Katchabaw, 2005; Charles et al., 2005). Thus instead of the system recommending related items, similar to online recommendation websites, the game will alter the gameplay or game content in order to fit the preferences of each player. Some games offer adaptive features that manipulate gameplay based on the player's skill level (Miller, 2004). There also exists a growing field of researchers that are building games that model player goals and traits in order to adapt to that particular player (Magerko & Laird, 2003; Thue et al., 2007; Togelius et al., 2007). Finding common ground between adaptive games and recommendation system can help influence how these adaptive systems will be produced in the future.

This paper argues that designing games that adapt to specific players will create challenging and engaging experiences for those players. One way to achieve this is by understanding how recommendation systems provide personalized content to users and how adaptive games can exploit those techniques. The author will begin by examining theories about creating situations where a person can have an engaging and/or challenging experience. This includes looking at how students learn within challenging environments and how challenging a person at their skill level can help them feel they are accomplishing something without becoming too bored or frustrated. Next, examples of how recommendation systems work within the web domain will be explored. These systems' techniques can help produce adaptive games that create experiences like those stated in the theories presented. Then, examples of game-related recommendation systems will also be investigated, consisting of systems that work outside of any specific game and contrasting them with examples of adaptive games. Finally, after comparing current recommendation systems and adaptive games, the author will focus on the missing recommendation techniques that adaptive games have yet to employ. Offering design suggestions, those missing recommendation techniques will be related to current games and shown how they can be used to enhance gameplay.

THEORETICAL FRAMEWORK

Players play games for different reasons (Sherry et al., 2006; Yee, 2007; Lazzaro, 2008). In theory, having an adaptive game that can cater to a user's specific reasons for playing will provide a better experience for the player than a game that has static gameplay. One of the reasons for playing, to face challenges, has been a target for adapted in games (Miller, 2004). Games inherently have conflicts or sets of challenges that players must overcome (Salen and Zimmerman, 2003). Facing challenges often means that a player wishes to improve their skills by taking on harder situations (Sherry et al., 2006; Lazzaro, 2008) and can include other gameplay reasons such as the need to discover, customize or become a good teamworker (Yee, 2007). The reason why some players want to face challenges within games can be related to theories that explore how people learn when faced with new challenges.

Lee Vygotsky's "Zone of Proximal Development" (Vygotsky, 1978) states that students should continually be challenged at the fringes of their abilities in order to promote learning. However, students need a certain amount of guidance before they can begin to venture off alone with any learning material. The "Zone of Proximal Development" is determined by how much help a student needs verses how much they achieve by themselves. This is similar to the concept of scaffolding (Instructional scaffolding, 2007) where students are given a large amount of support at the beginning of their development and slowly taken away as they master the material. Scaffolding has been used in other adaptive learning systems, called intelligent tutoring, in order to monitor learners as they progress through learning material (Anderson et al. 1995). Treating challenge as a reason for playing games, and following the concepts of scaffolding, an adaptive game system can observe how a player is performing in a game and act accordingly. This is exactly how dynamic difficulty systems operate (Miller, 2004).Thus, understanding how well a player is learning how to play the game can be used as a gauge of how much challenge a player should face in the game.

James Gee, following Vygotsky's work, outlined a series of learning principles that he felt can be found inside video games (Gee, 2007). Five of these principles are relevant to adaptive systems:

1. Achievement Principle – Learners receive achievements that reflect their skill level.
2. Practice Principle – Learners must do redundant tasks over and over but in an atmosphere that is enjoyable.

3. Ongoing Learning Principle – Learners must go through cycles of learning. The line between learner and master is vague, master must routinely change their behavior to adapt to the game's changing state.
4. "Regime of Competence" Principle – Learners operate on the edge of their abilities where they feel challenged but not overwhelmed.
5. Probing Principle - Learners probe the world, reflect in and on their actions, make a hypothesis and repeat.

The first principle, Achievement, provides learners with rewards and incentives to continue to push forward. The Practice and Probing principles give learners safe environments where they can continually test their skills without the fear of being judged. Finally the Ongoing Learning and "Regime of Competence" principles explain how games continually challenge learners and allow them to alter their skills with the purpose of becoming a master of a skill set, just like the "Zone of Proximal Development" states. The first four principles all share one thing in common, they are all dependent on a learner's skill level. What achievements are rewarded, what tasks must be done and the challenge put forth before the learner must reflect their skill level. Determining a learner's skill level is similar to determining their tastes or preferences in order to provide adapt gameplay and, as will be discussed later, how recommendation techniques can help adapt to players. Probing, the final principle, is an example of how a player's performance can be monitored. As players continually tests the game world their actions can be recorded and used to adjust the game's content accordingly which parallels how a recommendation system works.

Finally, as this section has discussed, adaptive games can deliver challenging experiences that hold just as much entertainment value as they do teaching value. However, not all players wish to continually face challenges and may want to play games for relaxation or to escape reality (Sherry et al., 2006; Yee, 2007). Adaptive games, for entertainment purposes, should be built to take these factors into account by monitoring how much of a challenge is needed to engage a player. Csikszentmihalyi's flow theory explains that an individual is in a "flow" state, or an engaged state, when they are experiencing a situation that is equal to their skill level (Csikszentmihalyi, 1991). This means that the activity they are performing is not so hard that they become frustrated but not too easy where they become bored. Similar to Gee's and Vygotsky's theories, flow theory looks at bringing a player to the edge of their competency level and continually challenging that level. In contrast, flow theory can also be used to keep users in a state that they find engaging and enjoyable without having to continually push them with new challenges. Adaptive games can use flow theory to determine whether or not a player wants to increase their proficiency in the game and can continue to provide for that player's desires.

Achieving a challenging experience is similar to achieving a challenging learning environment given the theories stated in this section. Also, understanding how to provide a challenging experience can be used to create a less challenging, yet engaging, experience if it is deemed necessary. In order to create these challenging and engaging experiences a game must monitor and make assumptions about the player of the game, adapting to their preferences. While recommendation systems do not traditionally challenge users, the next section will explore how recommendation systems monitor their users and, later in the paper, how these techniques can be used for adaptive games.

RECOMMENDATION SYSTEMS

Recommendation systems gather information from a specific set of information, filter the information (meaning finding useful patterns), and then output relevant results to users or the system's designers. These systems can be classified into three types of approaches: Content-based, Collaborative and a Hybrid approach (Adomavicius. & Tuzhilin, 2005).

- "Content-based recommendations: The user will be recommended items similar to the ones the user preferred in the past."
- "Collaborative recommendations: The user will be recommended items that people with similar tastes and preferences liked in the past."
- "Hybrid approaches: These methods combine collaborative and content-based methods."

Content-based approaches take an item-based and user-centric attitude while filtering. This means the output a user receives will be based heavily on the connections that exist between the items and themselves. An example of this is when Google Ad Sense determines the frequency of words on a webpage and delivers ads based on the most common words. In this case the user is the webpage and the items being recommended are the ads. In contrast, collaborative filtering takes a population-based approach to finding recommendations (both in terms of item population and user population). Looking at large populations collaborative approaches find patterns among the connections between every user and item and then relates a specific user to one of those patterns. These definitions are over generalizations of both content-based and collaborative systems but explain the principles of how each approach relies on different connection information that exists between users and items.

When a system has a low population a content-based approaches perform best, since they find connections between a single user and items (Segaran, 2007). Once a large number of users have contributed enough data to a system the collaborative approaches become much more useful (Segaran, 2007). Combining content and collaborative systems into a hybrid system can help alleviate the differences that exist between the two approaches (Adomavicius. & Tuzhilin, 2005). For instance, a weak example of a hybrid approaches is to perform a content-based filter when the system's user population is low, and then performing a collaborative filter as the population grows (Vozalis & Margaritis, 2003). The content-based and collaborative systems never touch one another, in this case, but greater integration of these two types of systems is possible (Adomavicius. & Tuzhilin, 2005).

Input, Filter, Output

The three phases of a recommendation system: input, filtering, and output, rely heavily on how a system sets up the connections between the system's users and items. These connections exist as three different types: User to User, Item to Item, and User to/from Item. These connections allow a recommendation system's filter to deliver outputs based on the connection patterns found. This sub-section will review each of these three phases and state how they compare and contrast to adaptive games.

Input

A recommendation system's information connections begins with the collecting of initial input data, for instance movie recommendation system's need movie information (genre, actors, and critic ratings) and user information (demographics, user ratings, and user actions). These information areas can be sorted into four categories which are also gathered by games:

- User Demographics – Contains any personal user information: age, gender, occupation, or user relation information such as a user's friends.
- User Opinions – The user's explicitly stated preferences about items or users. User rankings, user created categories, and user reviews would all fit into this category. Setting a game to easy, medium or hard is a small example of a player opinion found in games.
- User Actions – Actions that a user performs while using the system. These actions are generally implicit actions, not specifically asked to be performed, since the explicit actions that users can

perform fit in the demographic or opinion categories. Examples of implicit actions include how users navigate a websites or what music a user has in their playlist.. For games, how a player navigates the game's world or which quests they complete are implicit actions.

- Content Data – Any information that is used to describe items or content. Again, a movie recommendation system will describe a movie by its genre, the actors that star in the movie, when and where the movie was shot, etc. In games describing the properties of a race track or player items are examples that compose the game's content data.

There are three things to focus on when relating recommendation systems to adaptive games, given these categories of information input. First, content data and demographics work the same in both kinds of systems, both follow generalized information templates that define users or content, yet few games have begun collecting user created information from these areas. User-generated content (YouTube.com) and demographic / profile websites (Facebook.com, Myspace.com) allow users to contribute their own information for recommendation purposes, while games could benefit from collecting this type of information. Second, user opinions are not often found in games but are important in recommendation systems for delivering output. This may mean asking players opinion during or after gameplay could enhance adaptive games. Third, both recommendation and adaptive systems obtain user actions and process them in real-time (or as close to real-time as possible). Recommendation systems usually process output on-demand or in one time instances but adaptive games need to produce output in real-time and continuously. However, looking at ways that games can use single instance recommendation and faster ways to output recommendations continuously may expand how adaptive games operate.

In these ways adaptive games make use of similar data that recommendation systems would use to determine how a player's game should progress (i.e. a recommendation). How adaptive games, and recommendation systems, decide what recommendations to deliver to their users is by searching the connections that exist between the four categories of data above and applying filters.

Filter

Filtering is the heart and soul of any recommendation system. The search algorithms and the type of connections they search make up a system's filter, the process that will finally deliver the recommendation output. For example, a movie filter can recommend movies based on a user's movie rankings, age, and which movie genres they enjoy, each of which could be searched through separately or in various combinations. The different kinds of filter algorithms are too numerous to explain in detail within this paper. Instead a general overview of algorithm approaches and their different search techniques will be discussed.

There are two main types of filters, memory-based and model-based. Filters that are memory-based, meaning they have access to the input data (Vozalis & Margaritis, 2003), will use two types of connection or data comparisons: popularity and proximity. Popularity entails looking at how much weight is given to connections or data that occurs within, or between, items and users. Some examples include: Google Ad Sense looks for frequency of keywords within a document in order to display ads, a ranking website like Amazon uses explicit user ranking scores to find item popularity among users, and eHarmony.com, a dating website, connecting users who share common interests. Proximity comparisons look at how close data or connections occur between one another. One example is how Last.fm, a socially-drive music website, takes each of its user's music playlists and determines how often different recording artists are found together within the same playlist. Using these two types of comparisons similarities between users and items are able to be determined based on the initial input that is provided to a recommendation system.

Model-based approaches to filtering use methods to predict a user's preferences instead of reviewing the entire input dataset for every recommendation calculation (Vozalis & Margaritis, 2003). These filters will use the initial input data to train themselves to create a user models of each user. Once these models are created a system can make prediction how a user will act, such as how they will rank an item. Examples of model-based approaches include using Bayesian Networks and Monte Carlo algorithm methods (Adomavicius. & Tuzhilin, 2005; Vozalis & Margaritis, 2003).These model-based approaches have high overhead costs when they first compute the user models but these models provide recommendations faster, than memory-based filters, after they are create. A memory-based filter must consistently have access to the initial input data, which means it takes longer to retrieve recommendations but is always up-to-date. The model-based approaches only have access to the previously constructed user model based on the most up-to-date information the system had at the models creation time.

Games already make extensive use of model-based approaches when it comes to adapting gameplay to a player because they can produce faster results once a model is formed (Magerko & Laird, 2003; Thue et al., 2007). However, memory-based approaches, which use large datasets of input data, have not been implemented for adaptive gameplay purposes. These two approaches to filtering data will be discussed in the context of games in the next section.

Output

Each recommendation system's output is based on the patterns that the system's filters found when searching through, or creating a model from, the input data. These final recommendations can include item recommendations, user relationship recommendations, popularity rankings for items or users, etc. Different filters will affect what kinds of recommendation sets are available for output. A content-based approach will generally give recommendations based on similarities between items a user has previously ranked, while collaboration systems will compare multiple users to one another and provide recommendations based on their collective input.

One thing that differentiates a recommendation system from an adaptive game is that recommendation systems typically produce single instances of output and do not have to worry about ongoing interactivity. Adaptive games on the other hand can make use of their output to perform ongoing system behavior such as affecting the game's storyline events or altering how difficult a level will become. This means that adaptive game output can have repercussions well beyond the current state of the game or have to consistently create new output in order to keep up with the interactivity of the game.

GAME-RELATED RECOMMENDATION AND ADAPTIVE SYSTEMS

Recommendation systems and adaptive games have been shown to have similarities and differences:

1. Adaptive games can challenge users while recommendation systems help users.
2. Adaptive games rely on implicitly gathering user actions to adapt games and do not typically gather memory intensive explicit information (user opinions, demographics and user-generated content) while recommendation systems do.
3. Adaptive games must be able to deliver output to players in an interactive environment, continuously, but perform the same tasks as recommendation systems.

This section will look at current examples of recommendation and adaptive systems that are related to games in order to explore these differences further. These examples are split into two groups: systems that exist outside of gameplay (non-real-time) and systems that function as adaptive game systems while a game is running. As will be shown, the example systems that occur outside of gameplay take on the role of traditional recommendation systems while the in game examples will take an adaptive approach, making use of model-based filters to produce continuous output.

Outside Games

Game Review Websites

While not connected to any physical game, rating and review websites allow players to rate games and receive recommendations about which games may be worth purchasing. Amazon.com and other recommendation websites like itog.com, allow their users to rate games and will recommend other games based on these ratings. However, these websites are not strictly game-related and produce recommendations for plenty of other items. New Grounds and Gamespot are game-only review websites that allow critics and users to rank and review games. Game review websites take a collaborative approach to recommendation and use a memory-based filter to deliver a popular vote for games to the website's users. Game review and rating websites take a traditionally recommendation approach similar to other systems available on the internet. These websites do not have to worry about create recommendations in real-time but will provide some insight as to how memory-based filtering can be utilized for adaptive game design.

Matchmaking

Another category of game-related systems, that exist outside of gameplay, match players together to play games. Matchmaking systems pair together unrelated game players before they play a game. Xbox Live contains a matchmaking system which uses their TrueSkill player ranking protocol (TrueSkill (TM), 2007) which matches players together based on their skill ranking within each game. Matching players based on their skill level creates an evenly matched game which offers players a better experience. The TrueSkill system uses a collaborative approach where the performance of the player is compared against the other players in a single game after the game is over. This means players have no explicit way of affecting their rating besides how they perform within each single game.

TrueSkill uses a model-based type of recommendation filter that takes a player's performance during a game and uses it to create a predicted skill value for the player. However, a player's latest final game performance rating is the only value that is factored into their overall TrueSkill rank, thus previous ratings are never factored back into their current rank. In contrast to this model-based technique, recording how a player has progressed over a certain time period may help create better player data for adaptive games. Halo 3 is another example which matches players based on skill and experience.

Inside Games

Adaptive Difficulty Level Systems

Most games allow a player to choose a difficulty level at the beginning of the game. This choice is set on a single linear scale from easy to hard. A few games take away the need for users to make this choice at the beginning and instead adjust the difficulty level throughout the game. One concept that achieves dynamic difficulty is rubberbanding ("Rubberband AI," 2008) which affects the strength of each player throughout the game. A system using rubberbanding makes it hard for players to fall behind or get to far ahead in the game. For example, the game Mario Kart, a racing game, uses rubberbanding by giving stronger items to the players that are lagging behind in the race. These items will help the losing players catch up to the other players, thus affecting the competiveness of the game. This is an example of a collaborative system using a simple model-based filter, based on each player's current position, to determine how much help or hindrance each player receives.

For multiplayer competitive environments, e.g. sports or racing, with two or more players rubberbanding can be used effectively. Single player games, however, must use a player's performance based upon difficulty assumptions of the gameplay itself in order to create a competitive environment. Max Payne , a first-person shooter (FPS), is an example of a single player game where a player's information: health, accuracy, and number of kills, is recorded and used to adjust the game's difficulty. These records are compared against difficulty thresholds that are used to judge how well a user is playing the game (e.g. the player's accuracy was perfect in the last level so the game's difficulty will rise). This is another instance of a model-based approach within a content-oriented system where the player's most recent performance is used to determine the player's overall skill level (the model) for the next phase of the game (the content). This is supposed to create a flow situation where the user is constantly being challenged based on their skill level but never overwhelmed (Csikszentmihalyi, 1991). Other games that use dynamic difficulty approaches including, Sin: Episodes, Prey, and BioShock.

Player Modeling

Researchers are working on building adaptive systems that use player types (Bartle, 1996, 2003; Yee 2007) as a means to model players and alter gameplay accordingly (Magerko & Laird, 2003; Thue et al., 2007; Togelius et al., 2007). Three systems will be discussed in this section and each take a slightly different approach to gathering player data and creating internal player models. These player modeling systems are similar to content recommend systems using model-based approaches because they gather input from a single user and use their actions to form an internal prediction model of the player.

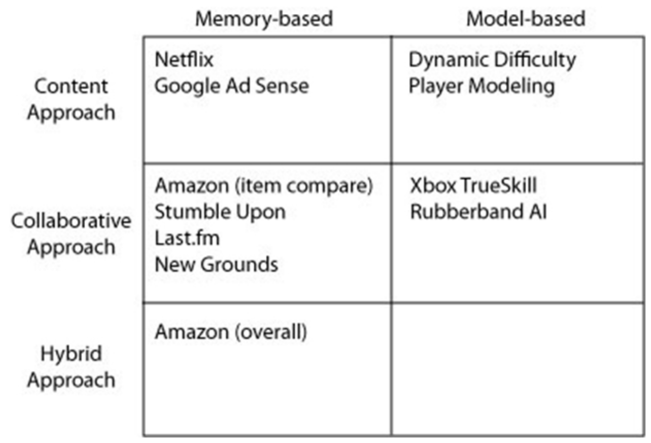
The Interactive Drama Architecture (IDA) was built to use a player model to help restrain the player from breaking the bounds of a game's story (Magerko & Laird, 2003). Interactive Drama is the concept of adapting a story to facilitate the preferences of a player, generally being achieved by using artificial intelligence (AI) to analyze a player's actions. IDA contains an AI Director agent which records the player's actions as they advance through the story. When the Director thinks the player is about to affect the story in a negative way (e.g. kill a main character, leave the playing area, or stall the story) proper actions can be taken in order to keep the story flowing (causing a distraction or providing a subtle hint) and moving forward.

A similar approach can be found in the PASSAGE system (Thue et al., 2007). Built on top of the Never Winter Nights engine, Aurora (BioWare, 2002-2008), the PASSAGE system also records a player's actions throughout a role-playing game (RPG). These actions affect different traits about a player that help determine what type of player is playing the game. There are five traits that a player is scored on: Fighter, Method-Actor, Storyteller, Tactician, or Power Gamer. Traits then determine what kind of quests the system will recommend and present to the player. For example, a player that chooses to always fight to finish a quest will be given future quests that focus mainly on fighting.

Last, a system built by (Togelius et al., 2007) adapts racetracks within a racing game. Each time a player completes a race a new track is procedurally generated based on the skill level of the player. The authors of this system found that optimizing tracks based on the player's skill were boring to players but that tracks were enjoyable when they were slightly harder than the player's skill level. As long as the player continues to play within the system it is able to create better track recommendations based on that player's skill level.

It should be noted that these systems do not allow players to affect their player model directly. Users of recommendation systems feel that the system works better when they have more control and wish the system to merely augment their experience, not completely take over (McDonald, 2003). It has been suggested that players be given a more explicit means of providing data in games as an additional way of gathering user input for adaptive purposes (Charles et al., 2005).

RECOMMENDING ADAPTIVE GAMEPLAY



	Memory-based	Model-based
Content Approach	Netflix Google Ad Sense	Dynamic Difficulty Player Modeling
Collaborative Approach	Amazon (item compare) Stumble Upon Last.fm New Grounds	Xbox TrueSkill Rubberband AI
Hybrid Approach	Amazon (overall)	

Figure 1. *A graph of which recommendation approach and filter each web-based and game system uses.*

Figure 1 graphs web-based recommendation and game adaptation systems based on each system's approach and filtering methods. The graph shows that while both types of systems take different approaches: content, collaborative or both, web-based systems focus on memory-based filters while game systems focus on model-based filters. This phenomenon occurs of the differences between adaptive games and recommendation systems: games need to produce challenges, games need to adapt in real-time and games do not collect memory intensive information.

Games need to produce extremely fast interaction results to provide real-time actions and challenges. Models will allow a system to avoid consistent checking and rechecking of input data, which would occur using memory-based models. Yet model creation incurs high overhead costs. Current entertainment games get around this fact by implementing limited modeling algorithms. For instance, the TrueSkill system and dynamic difficulty systems will produce player models only at specific points during the game or after its conclusion, and will only create collaborative models using small groups of players. Additionally models that continuously update themselves (i.e. the rubberbanding model) use simplistic information that can be quickly processed, such as race position, and do not use make use of memory intensive information. However, player modeling systems that are currently being researched have begun to go beyond these simplistic modeling functions to allow for more information to be gathered.

Web-based recommendation systems using memory-based systems are efficient in their own way. Connections between items and users can be easily computed and stored by employing databases to keep computation time down. While real-time computing is not as necessary for websites they do need access to accurate information, which is why memory-based filters are preferred. Under these circumstances web-based systems can relay recommendations back to the user that are up-to-date. Finally, the rare use of models for web-based systems may be due to the fact that models run the risk of over generalizing users and with most web-based systems containing thousands, if not millions of users, this could limit recommendation results.

While these two mediums, games and web-based systems, may have different goals in mind that does not mean they cannot learn from one another. Looking at games specifically, the notion of using memory-based filters seems to be non-existent (although game review websites use memory-based filters their goals are the same as other web-based recommendation systems, not games). Additionally entertainment games only make use of simple model-based approaches that deal with data from either a single player or a small group of players. Recent academic work has looked at expanding these models for single player use but large games, such as MMOs, could use model-based filters that work with larger player populations. Thus, reviewing ways that adaptive games can use explicitly gathered information (e.g. player opinions) with model-based filters and create models from larger populations may prove useful for creating adaptive games.

Games Using Memory-based Filters

Memory-based filters use large datasets that are collected over an extended period of time and contain the latest player and game information. These datasets could include information from the four information areas that recommendation systems gather. Examples of how games can use these information areas include examples that are already used by web-based recommendation systems: ranking content, profile matching, and tracking user actions.

Ranking Game Content

Ranking systems are the most common form of recommendation systems (Adomavicius. & Tuzhilin, 2005). Users are asked to rank items with a fixed number scale (1 to 5, 0 to 10, etc.). Each ranking score is recorded and linked to both the item and the user. Games rarely allow players to rank their experience in such a way. Building a memory-based filter where players rank a game's content and their experiences is a way to make use of both content and collaborative approaches for adapting a game based on player opinions.

First, ranking difficulty level could be used to help dynamic difficulty algorithms. Instead of the system judging the player's overall skill level the players themselves can state how difficult the game has been thus far. Second, players of massively multi-player online role playing games (MMORPGs) or single player RPG games could use similar tactics for determining which quests players will enjoy. After completing each quest a player could be asked to rank the difficulty, story aspects and the aesthetics of the quest. However in order to maintain agency within the game environment gathering ranking data would not have to be as explicit as displaying a ranking display similar to one found on Amazon or Netflix. For instance, various non-player characters (NPC) within WoW could ask the player for their ranking score while staying in character, asking questions relevant to the story. Third, games such as Spore rely on user generated content. Spore's developers have stated that users will be able to rank the user generated content built for the game (Shaw, 2008). In this case, each piece of content will be treated like an item found on an e-commerce website. In that way a collaborative recommendation system could be set up similar to those found on Amazon.com, Netflix.com or game websites like Kongregate.com. Games would automatically download content based both on overall ranking and how the aesthetics of the content meshes with the content that the player already owns. This would allow users to discover and download the available content that is of higher quality.

Profile Matching

Profile matching occurs on a number of different websites and is used for: social networking (Facebook.com), dating (eHarmony.com) and information sharing (Last.fm). These websites ask users to contribute demographic or content information so they may match the users together based on the data provided. Games can make use of profile matching both inside and outside of games.

Matchmaking does occurs outside of gameplay, for example the Trueskill system on Xbox Live (TrueSkill (TM), 2007). However, games could instead add the feature of matchmaking as an in-game mechanic and allow players to create profiles or match them along other game-related criteria. For example, World of Warcraft (WoW) allows users to announce that they are looking for a group to play with through various quests in the game. Though the system does not match players by their skill level (players on the same level will likely share interests), only by their need to join a group. However WoW does allow users to search the server for players that are around their level. If WoW used an internal profile matching mechanic it could match players, that need groups, together automatically based on their skill level or other criteria, similar to how Halo 3 matches players in-between matches.

Besides matching players together by skill level, a game could also allow users to have game profiles within the game. Players could use these profiles to create lists of their favorite content, allow for more role-playing options by giving them a personal section to write about their online personas, or give them a place to connect with other players. Furthermore, profile could also be made global, used by multiple games, which would help game portals or content delivery systems (websites or programs that allow access to many games, for example Valve's Steam system). These profiles would extend beyond any one particular game, becoming similar to online social networking profiles (e.g. Facebook and Myspace). Using profiles in this external way could be used to create connections between games and other interests a player may have outside of games, similar to websites like itog.com. For instance, a game profile that allows players to state outside interests could point "dog lovers" to virtual pet games like NintenDogs or NeoPets.

Action Tracking

Statistics tracking of player actions allows developers to review how players are playing their game and find any faults that may exist (Ludwig, 2007). This is also true for websites which use tools like Google Analytics to record how users are navigating and performing actions on a websites. Recording player actions was the key data collection method both for the adaptive and player modeling systems presented earlier in this paper. Current commercial games do record actions for gameplay purposes, for instance recording what quests a player has already performed, such as in World of Warcraft. This data is also used by some games that allow for extensive statistics tracking of player actions such as Steams statistics system (Valve Corporation, 2007). This type of elaborate action recording is used to gather information including: how long a game took to complete, which in-game characters are the most popular, or where players die the most often in the game.

This statistic information could be used to help recommend or adapt gameplay for players. Two examples used in games are 1) Dungeon Siege, a RPG, changes a player's character based on what items or skills they use frequently as they play the game and 2) Oblivion analyzes how a player behaves in its tutorial to suggest which class the player should choose for rest of the game. Another beneficial way of using player actions would be to combine actions from multiple players and make general assumptions about gameplay. If a system knows that 80% of players die in a certain area then the system could adapt to give a non-skilled player an easier time in that area. Other games which contain re-playable levels (e.g. Grand Turismo or Guitar Hero) could record how often a player plays certain levels and mark those levels as the player's favorites. If new downloadable levels (e.g. songs, race-tracks, etc.) are made available for these types of games then the system could also make recommendations based on these preferred game levels that the system has already determined.

Games Using Extended Model-based Collaborative Filters

The second way that games can utilize recommendation system functions is to gather data from a larger player population. Having access to larger datasets would allow games to create complex modeling filters that could generate player models from multiple players. Games that use model-based filter are usually limited lo single players or smaller populations, in the case of rubberbanding and matchmaking. Yet, model-based collaborative filters could connect player data together and form generalized player models using a game's entire population space.

StumbleUpon.com uses large sets of user profile data in order to recommend other users and items to users that have similar profile data and action patterns. This is similar to how the player modeling systems create models except they use much smaller player populations. Instead games could use this type of population information to create multiple player models that would be used to categorize players. A set of overarching player models could be achieved in this way allowing starting models to be applied to players as soon as they begin a game and can be refined as they continue to play. Even for a single player game, once one player model is created from a users play style it could be uploaded online and then be applied to other connected players who exhibit similar gameplay behavior. In cooperative multiplayer games player models could also be used to match players with other players of the same or complimentary models or, alternatively, players with conflicting models could also be matched together for competitive games.

In a larger scope player modeling could be used to create games that contain vastly different gameplay experiences based on each player's model. Many games already allow users to experience different gameplay throughout the game. For instance World of Warcraft designers have stated that their item auction system was designed to be harder to operate in order to force a player to decide whether they want to play the game as a trader or as an adventurer that spends most of their time away from the game's cities (Pardo, 2008). A more concrete example of different gameplay methods can be seen in games such as Natural Selection or Savage. These games allow one player to act as a commander who operates in the game world from an real-time strategy game perspective, while every other player plays the game like a first person shooter. These games allow the commander to be elected to the position but if a game used player models it would be able to recommend which players are best suited to be the commander (based on their past game performance perhaps). Furthermore a game could actually alter which game features are presented to users based on their player model. A fighter could receive more statistic tracking information or special moves that cause more damage, while a trader could receive better trading options or economic features. Player modeling could shift gameplay completely based on the analysis of how a player plays a game.

CONCLUSION

As it has been shown in this paper both challenging and engaging a user requires systems to monitor that user's preferences and skill level. Recommendation systems monitor users to help the system filter information that will be useful to the user. Adaptive games also monitor players to help the user find useful information but the information is used to provide a challenging or engaging experience in the game. Since recommendation systems and adaptive games both monitor users and filter information that will be given to those users they use the same types of methods and techniques when filtering and delivering output.

This paper began by looking at theories that state how challenging and engaging experiences are created for a user. One major factor in producing those types of experiences is by monitoring a user and altering the situations content to meet that user's needs. The author argues that both recommendation systems and

adaptive games can monitor and adapt to users. This began by exploring the methods and techniques that structure different recommendations systems.
Following an input, filter, and output information pipeline recommendation systems begin by collecting data in these four categories: user demographics, user opinions, user actions, and content data. Connections that exist between those data categories within a recommendation systems can be content-based (i.e. focuses on single user connections), collaborative (i.e. uses entire user population connections) or a combination. Additionally, filters within recommendation systems can take a memory-based (slower and accurate) or model-based (faster and generalizes) approach when searching for connections to finally output. Next, recommendation systems were compared to other game-realted recommendation systems and current examples of adaptive games.

While adaptive games and recommendation systems use similar techniques for filtering and output, they contrast in their goals, and therefore their characteristics. Games need to provide output that is in real-time and continuous. This means adaptive games usually make use of model-based filters in order to make recommendations quickly as to how the game should progress. Since memory-based filters are rarely used by games, player information: opinions, demographics and actions, are not stored long term and player actions are gathered implicitly as to speed up the information gathering process. Meaning a game's model-based filter must a) process simple information or b) gather information in small intervals throughout the game which is discarded once a new model is formed. Additionally, to order to provide continuous output, model-based filters only model single players or a small group of players at once within current games.

What this paper has shown is that while model-based approaches have been achieved and do provide challenging experiences for players, adaptive games have yet to seriously employ memory-based and complex collaborative model-based filters. Reviewing how these approaches are being handled by web-based systems new design suggestions for how they may be incorporated into games were discussed. Given that adaptive gameplay and recommendation systems follow the same principles, exploring these other methods may lead to new approaches to adaptive gameplay.

REFERENCES

Adomavicius., G. & Tuzhilin, A. (2005). Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. Knowledge and Data Engineering, IEEE Transactions on, 17(6), 734-749.

Anderson, C., Corbett, A., Koedinger, K., and Pelletier, R. (1995). Cognitive Tutors: Lessons Learn. The Journal of Learning Sciences, 4, 2, 167-195.

Bailey, C., & Katchabaw, M. (2005). An experimental testbed to enable auto-dynamic difficulty in modern video games. Paper presented at the GameOn North America Conference.

Bartle, R. (1996). Hearts, Clubs, Diamonds, Spades: Players Who Suit MUDs. Journal of MUD Research, 1, 1.

Bartle, R. (2003). Designing Virtual Worlds.

BioWare. (2002-2008). For Builders - Aurora Neverwinter Toolset. Retrieved February 19th, 2008

Blizzard Entertainment. (2008). New Interface Preview: Looking for Group. Retrieved February 19th, 2008, from http://www.worldofwarcraft.com/burningcrusade/townhall/lookingforgroup.html

Charles, D., McNeill, M., McAlister, M., Black, M., Moore, A., Stringer, K., et al. (2005). Player-Centred Game Design: Player Modelling and Adaptive Digital Games, Digital Games Research Association.

Chen, T., Han, W.-L., Wang, H.-D., Zhou, Y.-X., Xu, B., & Zang, B.-Y. (2007). Content recommendation systems based on private user profile. Paper presented at the Sixth International Conference on Machine Learning and Cybernetics.

Csikszentmihalyi, M. (1991). Flow: The Psychology of Optimal Experience.

Gee, J. (2007). What Video Games Have to Teach Us About Learning and Literacy: Palgrave Macmillian.

Herrmann, J. (2007). Got Gamers? Video Games: An engaging brand experience [Electronic Version]. Neilsen, Consumer Insight. Retrieved February 20, 2008 from http://www.nielsen.com/consumer_insight/issue2/ci_story1.html.
Instructional scaffolding. (2007, July 13). In Wikipedia, The Free Encyclopedia. Retrieved April 17, 2008, from http://en.wikipedia.org/w/index.php?title=Instructional_scaffolding&oldid=144464722

Lazzaro, Nicole (2007). The 4 Most Important Emotions of Game Design. Game Developers Conference, San Francisco.

Ludwig, J. (2007). Flogging: Data collection on the high seas. Paper presented at the Austin Game Developers Conference.

Magerko, B., & Laird, J. E. (2003). Building an Interactive Drama Architecture. Paper presented at the First International Conference on Technologies for Interactive Digital Storytelling and Entertainment.

McDonald, D. (2003). Recommending collaboration with social networks: a comparative evaluation. Paper presented at the Conference on Human Factors in Computing.

Miller, S. (2004). Auto-dynamic difficulty. Retrieved January 18, 2008, from http://dukenukem.typepad.com/game_matters/2004/01/autoadjusting_g.html

Pardo, R. (2008). Rules of Engagement: Blizzard's Approach to Multiplayer Game Design. Paper presented at the Game Developers Conference.

Rubberband AI. (2008, March 2). In Wikipedia, The Free Encyclopedia. Retrieved 05:33, March 2, 2008, from http://en.wikipedia.org/w/index.php?title=Rubberband_AI&oldid=188500815

Salen, K. & Zimmerman E. (2003). Rules of Play: Game Design Fundamentals. MIT Press.

Segaran, T. (2007). Programming Collective Intelligence: Building Smart Web 2.0 Applications. Sebastopol, CA: O'Reilly Media.

Sherry, J., Lucas, K., Greenberg, B., & Lachlan, K. (2006). Video game uses and gratifications as predictors of use and game preference. In P. Vorderer & J. Bryant (Eds.), Playing video games (pp. 213-224). Mahwah, NJ: Lawrence Erlbaum Associates.

Shaw, C. (2008). Pollinating the Universe: User-generated Content in SPORE. Paper presented at the Game Developers Conference.

Thue, D., Bulitko, V., Spetch, M., & Wasylishen, E. (2007). Interactive Storytelling: A Player Modelling Approach. Paper presented at the Artificial Intelligence and Interactive Digital Entertainment conference (AIIDE).

Togelius, J., De Nardi, R., & Lucas, S. (2007). Towards automatic personalised content creation in racing games. Paper presented at the IEEE Symposium on Computational Intelligence and Games.

TrueSkill (TM). (2007). Microsoft Research Retrieved Januaray 11th, 2008, from http://research.microsoft.com/mlp/apg/Details.aspx

Valve Corporation. (2007). Steam & Game Stats. Retrieved February 16th, 2008, from http://www.steampowered.com/v/index.php?area=stats

Vozalis, E., & Margaritis, K. G. (2003). Analysis of Recommender Systems' Algorithms. Paper presented at the The 6th Hellenic European Conference on Computer Mathematics \& its Applications.

Vygotsky. (1978). Mind in society: The development of higher psychological processes.

Yee, N. (2003-2006). The Daedalus Project. from http://www.nickyee.com/daedalus

Yee, N. (2007). Motivations of Play in Online Games. Journal of Cyber Psychology and Behavior, 9, 772-775.