Design Document

For

“The Breaking Point”

ICS 187 Computer Game Development

“The Breaking Point” and this document are designed by the Visions development team.

Mike Bartlett mbartlet@uci.edu
Gary Garon ggaron@uci.edu
Cathy Goings cgoings@uci.edu
Matthew Peronto mperonto@uci.edu

Table of Contents

1. Executive Summary

Written and Edited by the development team of Vision for use as a complete template in creating and editing "The Breaking Point". The purpose of "The Breaking Point" is to create a fun fantasy gaming experience for our users to enjoy.

In a distant land, long ago in the past, a race of men and a race of elves were at war. The war lasted for centuries, and as "The Breaking Point" picks up the story, there are few, if any, who remember why the war is being fought. Still, the soldiers fight blindly on, following the orders of their leaders to destroy their hated foes. The player will start the game in an elven fortress in the forest, held captive by the elven officer. He escapes with a precious elven sword, and the elven officer is tasked to hunt down the escaped prisoner, given the sword's twin to aid his quest. The pursuit lasts through the village and forests around the area, finishing with a climactic battle between the two officers on the border of the human/elven domains.

"The Breaking Point" game system will be a single-player, first person fantasy-adventure game. It will provide a 3D interface, through an engine designed from the ground up by the team at Visions. "The Breaking Point" will combine elements of first-person action games and RPGs, allowing the player to battle his way through enemies, while helping to create and build a sense of identity to his in-game persona. The player will be able to control his avatar throughout the game in swordplay and magic as will as giving limited tactical commands to 1-2 troops who will be supporting the character in the escape/pursuit.

As the player progresses through the game, he will be able to battle those trying to hinder his goal of escaping, as well as joining in and helping out in the ongoing battle around him. He will be able to do this through a skill-based combat system, in which the user selects a set of skills to use at any given time to strike, dodge, parry, and counter. Through the repeated use of skills, the player's character's skills will increase, and the continued defeat of his foes will help him become stronger and learn new skills and tactics. All of this combat is accentuated by the spoils of war, as the player can find better swords, armors, and shields to carry and equip themselves with as they move through the game. Between the exciting combat and immersive character design, "The Breaking Point" is designed to provide two hours of gameplay to complete once through, with additional gameplay time available to play through with different comrades, and completing different subquests.

"The Breaking Point" will be designed for Windows PCs, utilizing 3D Accelerator hardware through the DirectX API. It is slated for completion in mid-June 2003.

2. Application Context

<u>2.1 Rules and mechanics</u>

2.1.1 Characters

Each character, whether the player characters, a NPC, or an enemy, has the same set of statistics and skills. These statistics and skills are used to determine the results of combat, and the skills give different methods of attacking (and being attacked) to the player. Each character also has an inventory of items they are wearing and using (armor and weapons, described later). In addition, these statistics and the equipment are used to give each character a Life total (hit points) and an Energy total (magical power).

- *Statistics:* A character is made up of three major statistics. These are Power, Agility, and Spirit. Each statistic has a minimum value of 10, and a maximum value of 50.
  - Power – This is a measure of the character's physical power and strength. It is primarily used in determining the damage done by the character. It also has an effect on the chance to make a successful melee attack, and in determining the character's hit points.
  - Agility – This is a measure of how dexterous the character is. It is a measure of his ability to move and react quickly and precisely. It is primarily used in determining the success of an attack (especially ranged attacks), and also in the character's ability to dodge.
  - Spirit – This is a measure of how much innate magical ability the character has. It is used primarily to determine damage and ability of magical attacks, and also for how much Energy the character has for making magical attacks.

- *Determined Statstics:*
  - Life – Life is a measure of how much damage a character can take before he is killed. It is mostly determined by a base value based on the character type (human, elven, animal, etc.), but also is increased by strength. It is determined by the following formula (the base values will be determined later):
    Life = Base + (Strength / 2)
  - Energy – Energy is a measure of how much magical power the character has; basically it determines how many magical attacks the character can make. Each character type has a base, but Energy is mostly determined by the Spirit statistic, using the following formula (again, the base values will be determined later):
    Energy = Base + (Energy * 1.5)
  - Protection Rating (PR) – The value of the protection given by the armor of the character.

PR = Sum of all armor pieces

- Parry Skill – This is the character's ability to counter blows. It starts with a base value of 25, and can increase to a maximum of 100 as the character increases in skill with the given item, either the sword or the shield (see the "Leveling Up" section below for more on increasing skills). The formula for the parry skill 25 + (0.25 * Use)

- *Skills:* The skills a character has give him a variety of different attacks, counter-attacks, and evasion techniques to use when in combat. Skills also give bonuses to damage and attack success, or dodge, or parry chances. They are selected using skill sets, which are described in the Combat section below. In addition to the default skills (described below), there will be three special melee attack skills, three dodge skills, three counter-attack skills for each race (human and elven), and five magical skills. These will be designed later. The default skills are:

<u>Attack</u>

- Thrust – A simple thrusting attack with a sword. No attack bonuses.
- Swing – A simple downward swing with a sword. No attack bonuses.

<u>Dodge</u>

- Evade – Character darts to the side, then back to avoid a hit.

<u>Counter-Attack</u>

- Punch – When a melee attack is parried with a sword, the character can punch his attacker. +10 + (0.5 * Use) Hit-chance bonus. Fist base damage: 1-5.

<u>Magic: One handed</u>

- Magic Burst – One-handed. When making a Magic Burst attack, the character creates a ball of spiritual energy in his free hand, which is propelled at the target. Base Damage: (1 to 3) + (0.15 * Use). Cost: 3 Energy per attack

The extra skills will consist of:

<u>Attack</u>

- Ring of Blades – The character spins 360 degrees using a sword that causes damage to multiple enemies.
- Double Slash - This unique skill does damage depending on the number of enemies encountered. If there is solely one attacker, the character will slash twice; if there are two enemies, the character will first slash the direct attacker, then slash the other that is the nearest side attacker.
- Feint – The character does a quick pretend hit on the attacker, but then changes directions to surprise the enemy and swings sword to the other side of the enemy. This skill decreases the target's ability to parry.

- Cleaving – This is an overhead strike with the sword on enemies.
- Eleven Sword Special Attack – A destructive blow on enemies.

Dodge

- Jump – Character lightly jumps back to dodge a hit.
- Rolling – Character rolls to the side, which avoids the hit more quickly.

Counter-Attack

- Leg Swipe – Enemies are turned to the side by a quick leg swipe, and then a cleave (overhead attack) finishes off the enemies.
- Magic – As a character does a parry with sword, the character has time to use one handed magic to attack. This skill can only be used when no shield is equipped.
- Quick Dagger Reflex – When a character does a parry with a sword, the character quickly grabs a dagger from his inventory and uses it to quickly attack and offset the enemy. This skill can only be used when no shield is equipped.

Magic

No Handed Magic

- Heal – Provides some life to the character.
- Spirit Shield – If attacked, Energy can be substituted to take the loss of Life.
- Power boost – Increases the power stats for a short amount of time.
- Agility boost - Increases the agility stats for a short amount of time.

One handed Magic

- Drain – drains a small amount of life from the enemies and transfers it to the player.
- Rock launch – summons the rocks from the earth and hurls them at enemies in an arc.

Two handed Magic

- Area effect – hurts all enemies in the area of effect around the character
- Fire – Scorches all enemies, in a short arc directly in front of the player.
- Soul Reaper – an intense drain spell that does large amounts of damage (almost instant kill) to an enemy, and gives the Life back to the player.

2.1.2 Character Creation

When the player begins the game, he will find himself in the prison, as described in the story section. A character will approach and, along with some other brief dialogue, ask what his specialty is. At this point, the character description screen will open up. It shows that the character has started with the minimum of 10 in each of their statistics, but there will be 30 points available to distribute any way they wish among the three categories, along with an instruction directing the player to do so. Once the points are distributed, they will be directed to close the character screen, and their comrade will speak a line of dialogue based on their selection (either something like “A good fighter”, “Powerful in the spiritual arts” or “A jack-of-all-trades, huh?”). The character also starts with basic clothing, providing the minimal armor values as described in the Armor section, and with the default skills described in the Skills section. They start with no weapon, but that will be provided for them shortly in the game’s script.

2.1.3 Leveling Up

As the player progresses through the game, their skills will progress, making them better, stronger fighters. There are two main facets to this skill upgrading system. These are Skill-Use Improvement, and Milestone Improvements.

- *Skill-Use Improvement:* Each skill the player has will have a “use” value associated with it. Every time the character makes a kill on an enemy, the skill of each type that they primarily used will have its Use value increased by one. For example, if you attack once with the “Swing” skill, then try using the “Thrust” skill for three attacks, parry with your sword, and finish off the enemy with a “Punch” counter-attack, then the “Thrust” skill (since it was used more than the “Swing” attack skill), the sword-parry skill, and the “Punch” skill would all have their “use” value upgraded. The Use value is used in the bonus formulas for each skill, so the more a skill is used, the better the bonus it provides are.
- *Milestone Improvements:* After every 20 kills, the player character has reached a milestone, at which point they will get to upgrade their skills and statistics. At each milestone, the player will get to choose one skill from the available skills that they will now be able to use. In addition, they will get 5 points to be spread any way they wish to their Statistics (Power, Agility, and Spirit).

2.1.4 Equipment

All characters in the game, be they the player character, NPCs, or enemies, will be using certain equipment. This equipment includes armor, weapons, and shields. In addition, the player character will have an inventory (or backpack) in which he carries additional items that he is not currently using.

- *Armor:* There are two main features involved in classifying armor. The first is the body location for the armor. There are six body locations that can be armored: Head, Body, Arms, Hands, Legs, and Feet. The second classifying feature of armor is the type. The types range from leather armor, to studded leather armor, on to mails, and as high as plate armors. Each body location has a minimum clothing/toughness value associated with it (shown below) which adds to the character's Protection Rating, and each type of armor provides a "bonus" to that value for the location on which it's worn. In addition to the larger armor bonus each higher type of armor provides, each stronger armor also limits your character's mobility, decreasing the chance to make a successful attack. Following is a list of the minimum values, and the values some armor types add to it.

	Minimum	Leather	Scale Mail	Plate Mail	Maximum
Head	3	5	10	12	15
Body	6	10	18	24	30
Arms	4	7	12	15	20
Hands	2	4	6	8	10
Legs	3	5	10	12	15
Feet	2	4	6	8	10

Attack Chance Decreases: Leather: -0, Studded Leather -1, Ring Mail –5, Scale Mail, -10, Breast Plate –8, Plate Mail – 15, Full Plate Mail -20

- *Weapons:* There are two types of weapons in the game, which are swords and bows. However, there are only two types of bows; these are a short bow, and a long bow. The short bow is slightly faster, but does less damage and has less range. The long bow is slower, but does more damage and has slightly longer range. There are more types of swords, however. Each type of sword does more damage as they get larger, but some take longer to swing than others. In addition, some swords may provide additions to the chance to make a successful attack based on their design, and there is one sword, the hero's sword, which also provides a bonus to the player's magical attacks. These bonus-granting swords are just variations on the types of swords shown below, such as "serrated" or "sharp" or "runed". The speeds below are multipliers against what the actual sword swing speed is (to be determined later in balancing and animation). In addition, the hero sword below adds 20 to the chance for a successful attack, and adds 25% to magical attack damage. However, since it is an elven sword, it has negative effects on a human wielding it, causing the loss of 1 Life every three seconds.

	Damage	Speed
Dagger	1-4	0.5

Short Sword	1-6	0.7
Long Sword	2-8	1.0
Bastard Sd	4-12	1.5
Hero (Long)	3-10	0.9

- *Shields:* There will be three different shields throughout the game. Each type of shield provides more protection. The disadvantage to using a shield is that the counter-attacks available aren't as powerful as those from a sword-parry.

	Protection Increase
Small Shield	5
Large Shield	10
Tower Shield	15

- *Healing/Recharging:* As the player progresses through the game, they will use their Energy for spells, and lose Life as they take damage from enemies. In order to refill their Life and Energy, the player will tap the spiritual energy from departed characters (enemies and allies alike). As they walk up to corpse, the user can push "M" to refill some of their Energy from the corpse, and "L" to refill their life. Each character has a certain amount of spirit to be tapped, so once it is tapped, it cannot be tapped again. This includes any tapping done by "drain" skills (if the player uses a "Drain" skill on a foe, then they cannot tap additional life or energy from that enemy's corpse).

- *Inventory:* In addition to the equipment he is wearing, the player character will be able to carry other gear. This extra gear will be stored in the player's inventory. The inventory will also allow the player to manage which equipment they are currently wearing. The inventory will allow the player to hold up to 18 items; these include the 8 equipped items and 10 additional items.

2.1.5 Dialogue and NPC Interaction

As the player moves through the game, they will be able to interact with other characters in the game. To talk to a character, the player must walk up to them and use the "Use" function. This will initiate dialogue mode. In dialogue mode, the NPC's dialogue will be displayed and the sound file (if there is one) associated with that dialogue line will be played. Then a varied number of options will immediately be given to the player to be selected (before the sound file has stopped playing). Depending on how the conversation advances, NPCs can give advice regarding the current mission, give quests to the player, and even give the player an item in some cases. Each time the player is given a selection of dialogue options; the last option will always be the one to leave the conversation, allowing them to stop talking to them at any time. This is essential, as the game will not be paused during dialogue mode, and an attacking character may approach while the player is talking to a NPC.

2.1.6 Movement

The game will allow the player to move, in real-time, in a 360 degree zone. The character will have two movement speeds: “walk” and “run”. As the goal in this game is to be making a quick escape, and moving quickly in combat, the character will be set by default at his “run” speed (the actual speed of the walking and running will be determined later in the design process). While the freedom of motion is relatively large, there will still be edges of each zone that the player will not be able to walk past (walls in the prison and town, tree barriers in the forest, etc.).

2.1.7 Combat

Combat will also happen in real-time. There are two different types of attacks: Melee (hand-to-hand, sword combat) and magical. Each type of attack can be accentuated by the character’s stats, items, and by the use of one or more of his skills.

- *Melee Combat:* The basics of melee combat involve closing to close range and using the “attack” function to make an attack with the player’s sword. Each attack will cause the player model to undergo an attack animation, and will cause the game to calculate the result of the attack. Each skill has associated with it an “attack zone”, consisting of a “range” (also influenced by the weapon) and an “attack arc”. If there is an enemy target in range of the character and within the attack arc, then an attack can either result in a “hit,” a “miss,” a “dodge,” or a “parry”. This result is determined by a series of random numbers, which is compared to a series of “target numbers” that are calculated by comparing the player’s relevant statistics with the target’s relevant statistics. Results are checked in the following order: Hit/Miss, Dodge, Parry. If an attack is parried, the defender may have a chance for a “counter-attack”. If an attack is not in the valid attack zone for the skill, it automatically misses. If there are multiple targets in the attack zone, then the one closest to the center of the screen is the target.
  - *Selecting which skills to use:* The player will be able to set up to five “skill sets” for melee attacks. The player can switch between these sets through hotkeys. Each skill set will include a chosen Attack skill, Dodge skill, Parry skill, and Counter-Attack skill. When the player initiates an attack, the attack skill of the current set is used. When the player is attacked, it uses the dodge and parry skills of the current set when calculating the target numbers for each of those events, respectively. If the player has a chance to make, and in fact attempts a counter-attack, the counter-attack skill of the current set is the one that is used for that

attack. For NPCs and enemies, they will have a single default skill set (the skills of which will be different for each type of character), which will always be used when handling their melee combat.

- *Damage:* Damage is determined based on the statistics of the weapon the character is using for the attack, modified by the character's Power statistic and any skill bonuses they may have, as shown by the following formula.
  Damage = Weapon Base * (1 + (Power / 50) + Skill Bonus Percentages)
- *Determining a hit or miss:* To see if an attack hits the target, the game uses the following formula to calculate the Target Number (TN):
  Attacker's Skill (AS): (Attacker's Agility * 2) + (Attacker's Power) + (Attack Bonuses)
  Defender's Skill (DS): (Defender's Agility * 2) + (Defender's Protection Rating)
  TN = (AS + 100) / (AS + DS)
  The game then generates a random number (floating point) from 0-1. If this number is less than or equal to the TN, then the attack hits, otherwise, it misses. If an attack hits, the game will then check to see if it is dodged or parried.
- *Determining a Dodge:* To see if the target dodges a successful attack, the game uses the following formula to computer a TN:
  TN = (Defender's Agility + Defender's Dodge Skill Bonuses) / (Attacker's Agility + Attacker's Accuracy Skill Bonuses + Defender's Agility + Defender's Dodge Skill Bonuses)
  The game then generates a random floating point number from 0-1. If this number is less than or equal to the TN, then the attack is dodged, otherwise.
- *Determining a Parry:* A parry involves blocking the attackers swing, whether with a sword or with a shield (which of these is used is set in the character's current skill set). Seeing if the target parries the attack works exactly like a hit-check, or a dodge-check, only with the following TN formula:
  TN = Defender's Parry Skill / (Defender's Parry Skill + Attacker's Parry Skill)
- *Counter-attacks:* A counter-attack, if the defender has an appropriate counter-attack skill, can be made when an attack is parried. A counter-attack uses the same TN as a regular attack, with the counter-attack skill bonus included in the "attack bonus" part of the formula. A counter-attack can be dodged, but cannot be parried (consider that the previous attacker's weapon may be "out of position" because his attack was parried, but he still may be able to dodge the counter if he is agile enough). When an attack is parried, an icon will appear in the corner of the screen indicating the player has the ability to make a counter-attack. The player must then initiate an attack within two seconds, at which time the icon will disappear, in order to make a counter-attack.

- *Magical Combat:* The majority of ranged combat will be done with magical attacks, though magical attacks may also involve short(er) range skills. To initiate a magical attack, the player will use the "secondary attack" function. This will cause the animation for the currently selected magical skill (see below for the description of selecting magical skills) to start, and for the magical projectile (if any) to be fired. The projectile will then proceed to the target (either homing on the target currently under the crosshairs, or in a straight line, as outlined in each skill's description) until it strikes, reaches its range, or reaches the point where it is supposed to initiate some effect (such as explode). Magical skills are limited, however, by the number of hands the character has available to use in casting them. Some spells require both hands to be available (no sword or shield equipped) to cast, some require one hand to be available (a sword, but not shield), and some spells (generally enchantment or shielding spells) may require no hands available (a sword and a shield can be worn).

  - *Selecting Magical Skills:* Since the player will likely have multiple magical attacks they may want to use, unaffected by what melee attack skills they are using, Magical skills are selected by different hot-keys than melee skills. The player can have a set of hotkeys for up to three different magical attack skills for each number of hands they have available to cast the spell (one, two, or none).
  - *Determining Hit-or-Miss:* A magical attack always hits if it strikes its target.
  - *Damage:* The damage for a magical attack depends on the skill used, and it is enhanced by any bonuses given by equipment worn, and also by your Spirit ability.

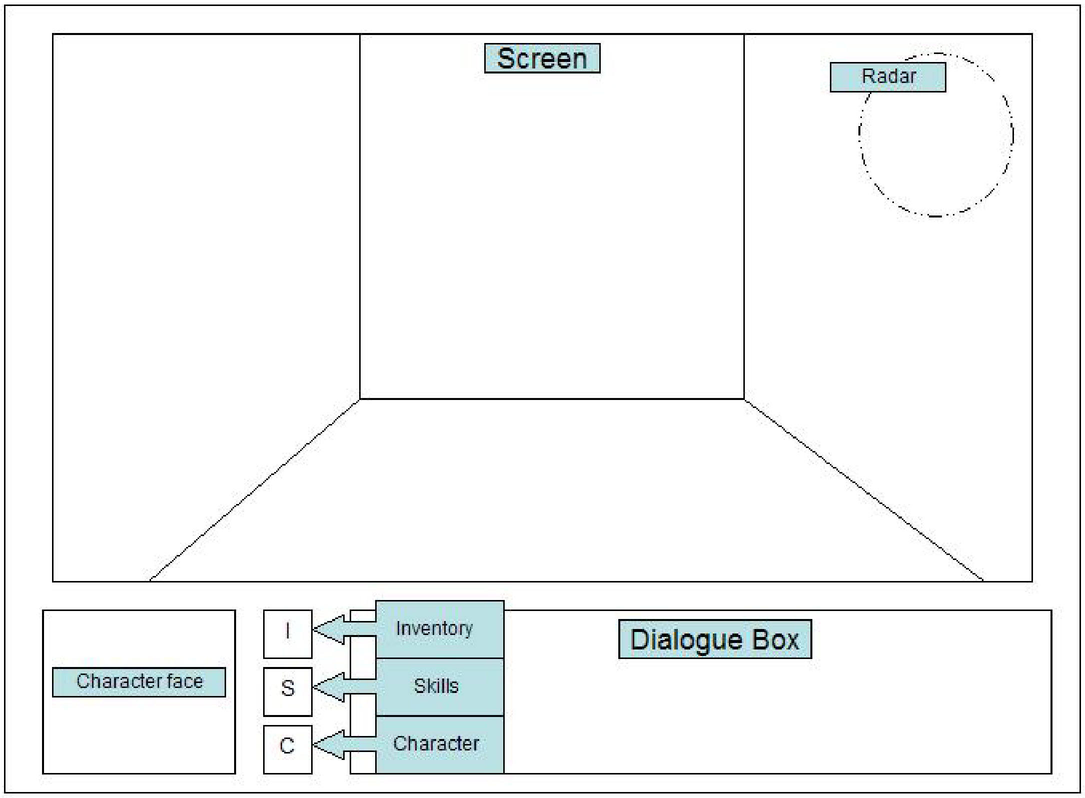
    Damage = Base Skill Damage * (1 + (Spirit / 40) + Weapon Magic Damage Bonuses)

2.1.8 Dying

As the goal of the player is to kill his enemies while making his escape, his opponents have a similar goal to kill him. As such, there is a good chance the player may be killed in combat. Since the game is relatively short and there isn't an in game save system, in order to keep the "fun" factor of the game up, we won't have any serious death penalty inflicted on the player; instead we will have "checkpoints" within each zone. These checkpoints won't be visible to the player, but will be certain distances or locations through the level which, when passed, will be where the player will restart after dying. When restarting the player at one of these points, enemies will be kept a safe distance away (far enough away that the player will be given a little bit of motion without attracting the attention of any foes), and any enemies the player was fighting will be restored to full health. His comrades will also restart at the checkpoint with him. From here the player will then continue his quest to escape, throwing himself back at the enemies that killed him before, hopefully now with a new strategy to keep from dying again.

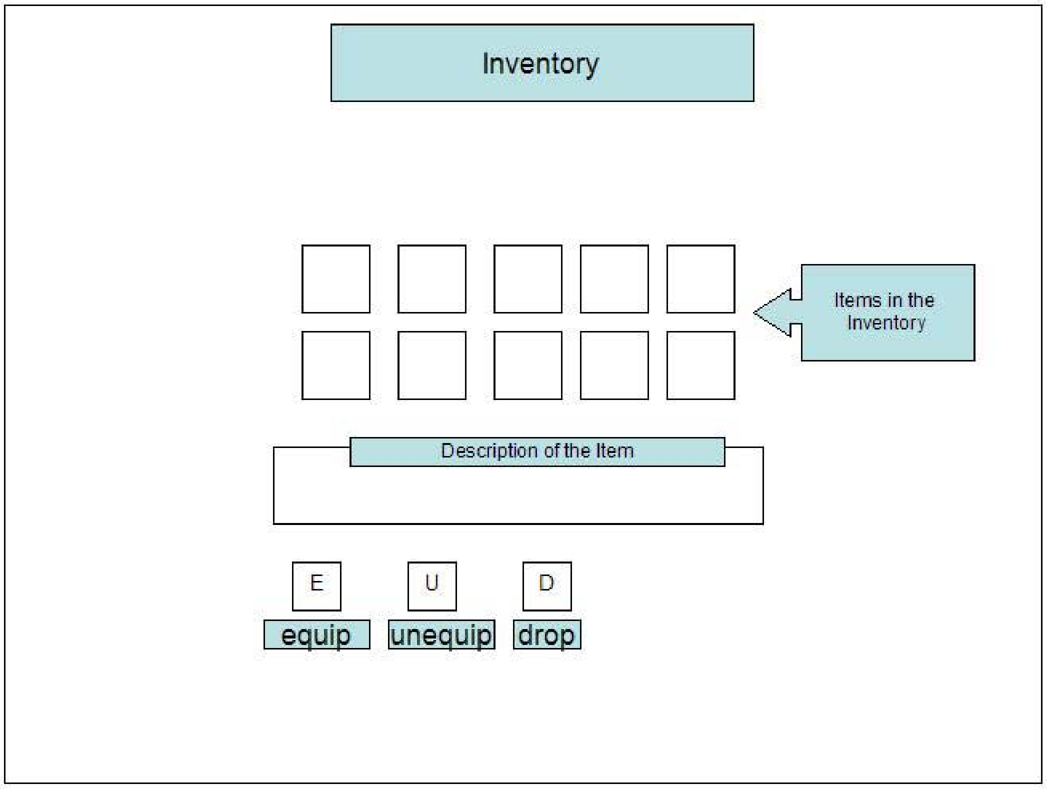
<u>2.2 Artwork and user interface</u>

2.2.1 Sample Screen Shot



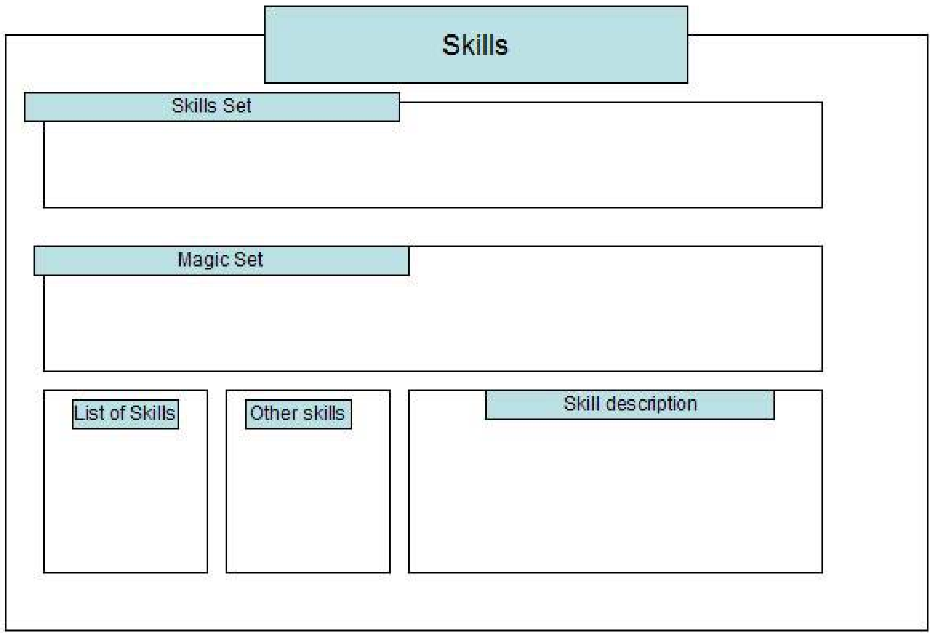
This is a sample screen shot of what the user will be seeing throughout the game place. There will be a radar to the right to show the player where enemies lurk, and where the character’s comrades are located. The radar will be useful for combat. The bottom task bar will show the character’s face, and menu button options that can access the character’s inventory, character’s set of skills, and the character’s description. Each of these buttons will open up their own individual screens. A dialogue box will be located on the bottom right.

2.2.2 Inventory Screen Shot



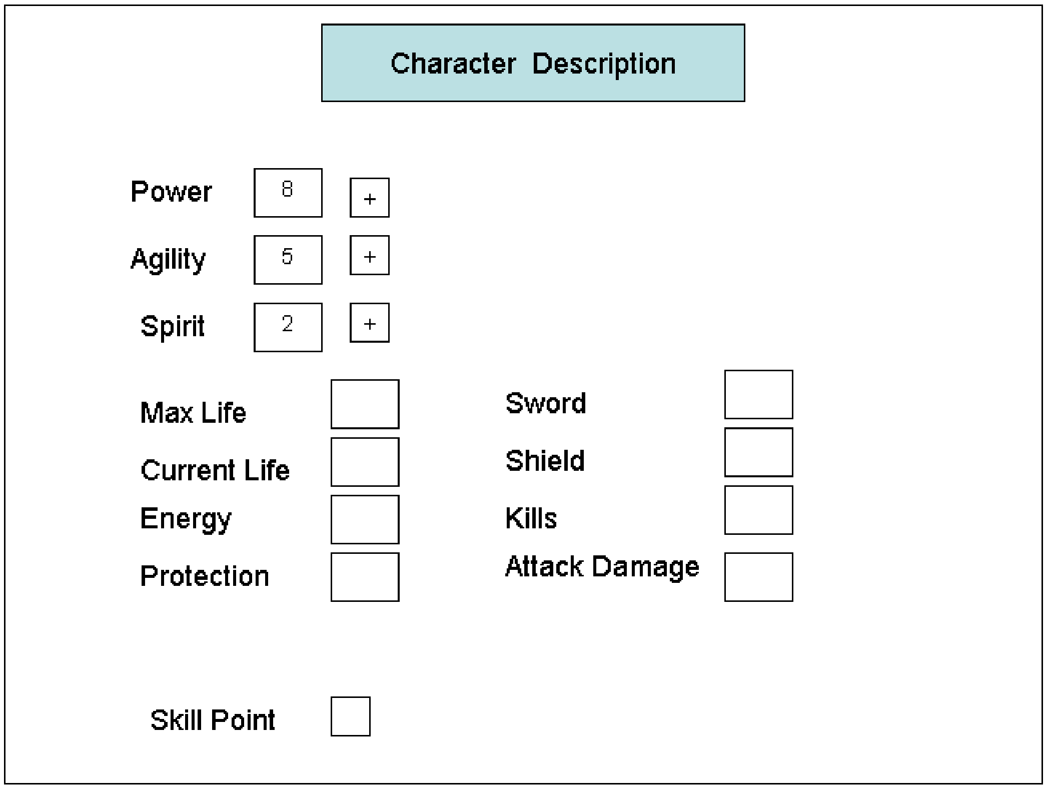
These items will be displayed as square graphical icons on the player's inventory screen. Equipped items will have the background of their icons tinted blue, while non-equipped items will have a black icon background. Also on the inventory screen will be "Equip", "Unequip", and "Drop" buttons. In order to equip an item, the player will select an item (selecting an item in the inventory will turn the icon background white), then push the "Equip" button. Items can be unequipped or dropped in a similar fashion. If the player tries to equip an item when they already have another item of that type equipped (i.e. they try to equip a chain mail body armor when they are already wearing a leather body armor), then the currently equipped item will be unequipped, and the new item will be the one that is worn.

2.2.3 Skills Screen Shot



The skill set-up screen will be accessible by pressing the “L” key, or by pressing the “Skills” button on the main interface screen. Each skill will be represented by its text-title, not with an icon like in the Inventory. The skill set-up screen will have three main parts. The first part, at the top of the screen, will be the “Skill Set Selection” dialog. This is where the player will be able to set-up the skill sets they will use in combat in the game (as described in the combat section). The second part of the screen, in the middle, is the “Magic Hot-Keys” section. This, like the Skill Set Selection, allows the user to set-up which magical attacks will be bound to their three magic attack hot-keys, depending on how many hands they have available to use magic. The final part, at the bottom of the screen, is the skill listing and description section. This section will include, on the left, buttons for each of the skill types (Attack, Dodge, Counter, Magical), a listing of the skills of the currently selected type directly to the right of that, and a description of the currently selected skill to the right of that. The listing of skills for each skill type will include ALL available skills for that type, even those that the player’s character hasn’t learned yet. Skills the player’s character knows and can use will be listed in bold, and skills the character has not yet learned will be in italics. In order to set a skill in the skill set selection, or the magic hot-keys section, the player first must click on the skill in question in the skill-set selection section (for example, they must click on the Attack skill in the Second Skill Set). Once they have done this, the corresponding skill type will be listed in the right section of the screen. The player simply needs to click on the skill they wish to use in order to add it to the skill set, or set it as a hot-keyed magical attack. If the skill the player selects is not yet learned, or is not a valid selection for that slot (for example, selecting a two-handed magical attack skill when they are trying to pick a one-handed magical attack), then no change will be made and everything will be deselected, starting the selection process over.

2.2.4 Character Screen Shot


Character Description

Power	8	+		
Agility	5	+		
Spirit	2	+		
Max Life			Sword	
Current Life			Shield	
Energy			Kills	
Protection			Attack Damage	
Skill Point				

The character description screen will contain all the basic statistics and attributes of the player's character. It will be accessible either by pushing the "H" key, or by pushing the Character button on the main interface screen. The screen will show the player the character's three main Statistics' values (Power, Agility, and Spirit), as well as their Maximum and Current Life and Energy, the character's Protection Rating, Sword and Shield parry skill values, and the number of kills made. It will also have the damage and Attacker skill (a rating of their relative ability to hit enemies) when using the currently equipped weapon and using the regular attack skill of the currently selected skill set. When the player's character reaches a Milestone, a box displaying how many statistic points the player has to distribute will appear, as will buttons next to each of the three major statistics. Each push of the button next to a statistic will increase that statistic by one, as long as the player still has statistic points to use.

<u>2.3 Gameplay and balance</u>

"The Breaking Point" is an Action/Adventure RPG, and as such, it combines elements of real-time combat with character building and NPC-interaction. The goal of "The Breaking Point", as described in the story section, is to move through the story of an escape from a prisoner-of-war camp, either as the human prisoner escaping, or as the elven guard officer chasing him. Doing this will require multiple interactions with the game world.

2.3.1   Keeping track of your character

As the player's character increases in skill and finds new items, it is imperative that they can keep track of all the character information. A screen will be provided for character statistics, which will list information on the character's statistics, life, energy, and protection information, and where the player can distribute attributes gained through experience. There will also be a screen for managing skills. This screen will list all skills that are learned, all skills that can be learned, and will allow the player to set-up the skill sets to be used when in combat. Finally, there will also be a screen for managing the character's inventory, which will allow the player to select which items are currently in use, and to store and move around items in the character's back-pack. Unlike while browsing the character and skill setup screens, where the game is paused, while browsing the inventory screen the game will NOT be paused.

2.3.2 Moving through the game world

Movement will be available to the player in real-time, with 360 degrees of movement freedom. Movement will, by default, be accomplished by a mouse and keyboard combo, with the mouse providing the look/turn left and right, and the keyboard providing forward, backward, and sideways motion in the direction the mouse has the user looking. This is similar to the set-up in most first-person shooter games. The player will be able to move through the prison and town, into certain rooms and houses, and then out into the forest, where there will be Zones in which the player moves. Each zone will have a barrier of trees along the edge, which the player cannot pass. However, there will be certain openings in the tree barriers which, while the user cannot pass through them while in a zone, as they approach them they will be given the option to move through the opening into another zone, by using the Use function (described in “Interacting with the game world” below)

2.3.3 Interacting with the game world

As an RPG, there are different ways to interact with the game world. There are three main interaction functions for the player as they progress through the game. First is the “Attack” function. This can be used on any character in the game, even friendly NPCs. The actual action taken when using the attack function depends on the skill set currently selected. Secondly is the “Secondary Attack” function, which behaves similarly to the regular Attack function, except it is used for magical attacks instead of physical (ranged/melee) attacks. Finally, there is the “Use” function, which allows the user to interact with certain objects in the game. The Use function allows the user to open certain doors, initiate dialogue with NPCs, and to move between zones when prompted.

2.3.4 Progressing through the game world

The goal of your movement through the game world, even though it allows 360 degrees of motion, is to keep moving “forward” through the zones, away from the elven prison and town, and toward the human base. If the player is playing as a human, then after a certain amount of time, guards will approach (spawn) from the direction of the prison, spawning further and further into the woods the longer the player is in the woods. If the player is playing as an elf, then there will be more human reinforcements coming from their base and getting between you and the escaped prisoners. Moving into side zones will allow the player to get more experience and increase their skills, but at the cost of making it more difficult to get to the end. The main goal of moving into the side zones is when the player is given a subquest by a NPC. Most often, the subquests will involve moving into one of these side zones in order to complete them. The subquests will provide the player with more chances to increase their character’s skills, and also provide rewards of better weapons and armor, making moving into the side zones a better trade-off for the ground lost against the pursuing forces.

2.3.5 Fighting with your friends

Some of the battles in the player’s escape may find him outnumbered greatly by his enemies. To help out in these tough scrapes, the player will have the option of having two comrades follow him in his mission. These two characters will follow the player’s character wherever he goes. They will not be directly controllable (i.e. they can’t be commanded to “Attack this target”), but the player will be able to give them basic orders either through hot-keys, or through a comrade-orders menu. The orders will include “Attack”, “Follow Me”, “Defend

Me", and "Run Away". When they are given an order, the comrades will fall into an AI script that helps them determine how best to carry out these orders (i.e. who to attack, where to move, etc.). In addition, the comrades may have optional, but greatly beneficial (to the player's and the comrade's abilities) subquests to give the player upon joining up with him.

2.3.6 Completing the game

Once the player has made it through the forest, destroying or evading their pursuit, and helping their side's cause in the war by completing subquests and killing enemies, they will finally be caught by the opposing hero in a grove just outside the human base. In order to complete the game, the player must defeat his counterpart in single combat. The hero will be at a set skill level, so the less skilled the player's character is relative to that (the less time they spent on side zones and subquests), the harder the final battle will be.

2. 4 Music and sound

Music and sound will be from various MP3s or CDs of team members. Background music in the game will depend on the player's location and setting in the game. For example, battle mode will have a much faster, upbeat rhythm, while roaming around the forest will play a pleasant tune. Also, the ending battle of the game will have a certain distinctive song. Sounds such as sword clashing will be taken from various CD sources or recordings. Characters that will need voice will be done using voice over, voluntarily done by various members of the group.

2.5 Background story

Prologue

In ancient history there was peace throughout the land. Elves and humans lived together in a perfect harmony. A race of creatures known as the spirits lived alongside the other two races. The land was fertile and without war. Every time a living being passed beyond this realm a portion of his life essence would become a part of the land.

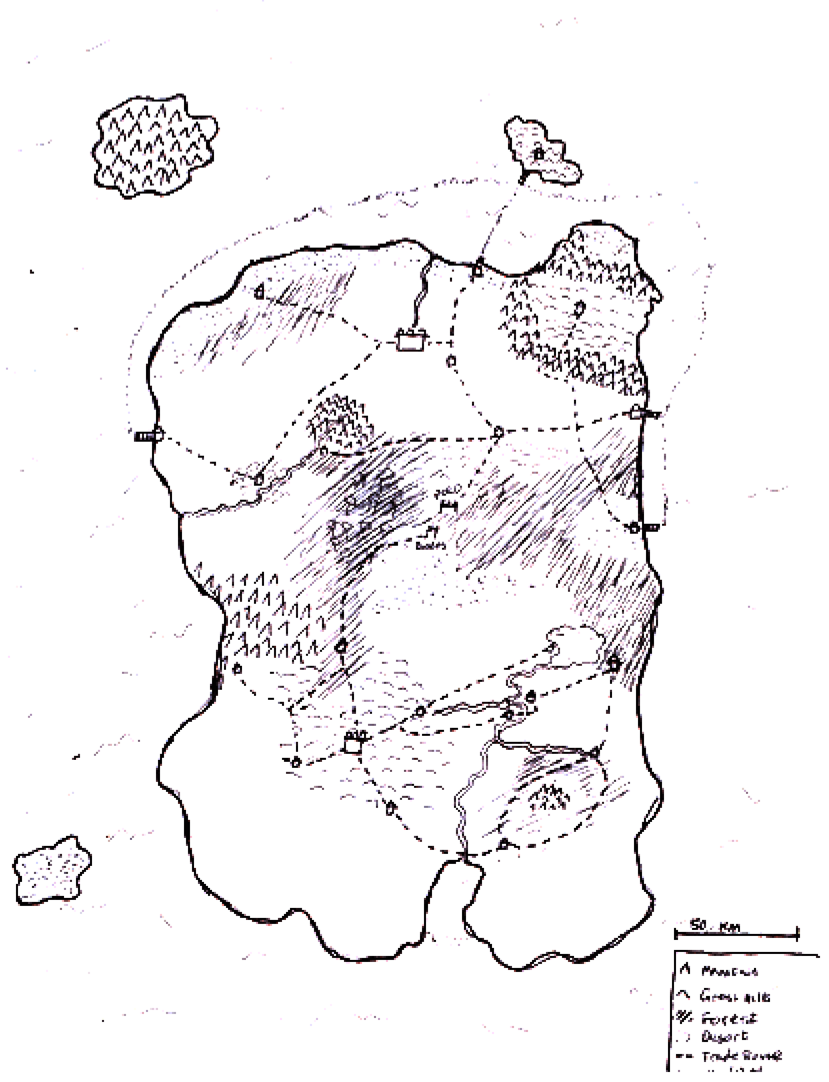
The spirits had the ability to shape the land for the betterment of life amongst the civilizations. One such spirit decided that elves and humans should be able to shape the land as they pleased. She trained a single human and elf in the art of using the life essence of one passed to help the land.

This human and elf both went their separate ways to help the civilizations of their people. As time progressed both the elf and human realized that using the essence of life for the good of the land was keeping their personal life essences strong. Out of selfless desires the elf and human both began to train their people in the powers the spirit had taught. The civilizations expanded beyond what any living being had imagined. But only the two taught by the spirit continued to live.

The problem was that power brought with it corruption and greed. The humans and elves began an age of war. As the life essence was used for selfish purposes the land began to fall and with it the spirits. After more than a century the elf and human endowed with the power granted by the spirit realized they were in a desperate situation. They both approached the spirit demanding she fix the problems with the world. She tried to explain to them that it was their own greed and selfish acts that was causing the world to diminish. Unwilling to hear this both the elf and human blamed the spirit for destroying their people and their world.

With this act the spirit, the last spirit, perished. With her last breath she cursed the elf and human to die and be reborn through every age until their civilizations could once again live selflessly. With the last spirit gone the elf and human aged through the years they had bypassed in a matter of moments. The human civilization upon realizing their leaders demise blamed the elves for destroying their chance at eternal life. Likewise the elves blamed the humans for ridding the world of the spirits.

War began. The humans and elves fought through centuries to come. They continued to use the essence of life as they were taught through the spirits, but now for the purpose of war. The more selfish acts they committed the more they destroyed their own world. The elf and human were born again and again to shape the world, only to continuously destroy each other. This is the story of one rebirth of the elf and human. The elf is reborn as Danongale, the human Miotis.



The Story

The land is known as Mandonia. Elves and humans have struggled for centuries to control the resources in the overpopulated land. Once there was a peaceful existence between elves and humans, but in ancient history the peace was destroyed. No living soul knows why the struggle began, but all know that the need to fight has become a part of life.

Every child, elf or human, begins fighting once they are old enough. This has been done for generations and is continually done by tradition. The same is such for the human sergeant Miotis and the elf lord Danongale. Miotis joined the human's "King's Army" as expected. With his natural ability for the arcane arts, he has advanced quickly through the "Kings Army". Until recently, he was stationed in the human city known as Ford.

Danongale is an elf lord by birth. Because of his lineage he has gained much respect within the elf army. His respect is well earned though. Through numerous battles and conquest Danongale has been placed in charge of the shrine swords in the elf city Dietos.

The cities of Dietos and Ford both lie close to the border of the human and elf territory. A dense forest separates the lands here. However, the forest thins at a point known by both elves and humans as the Breaking. The two cities lie within a mile of the forest on either side of the Breaking.

Miotis, while fighting within the forests of the Breaking, was captured by the elf lord Danongale. Miotis is currently being held prisoner by the elves in their town Dietos. The story begins as a band of humans break Miotis free from the elf prison. Control Miotis in the chase through the breaking as Danongale attempts to capture the escaped human sergeant.

<u>2.6 Characters</u>

Human Hero: Miotis

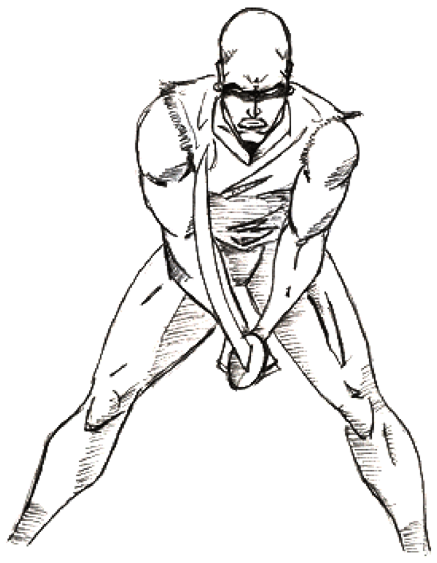
Miotis is a sergeant in the King's army. He is a human, blessed with the a strength in the spirit power. He is the reincarnation of Danongale's counterpart, destined to an unending life and battle. Born again in the world Danongale quickly scaled the ladder of power within the king's army because of his natural talents. He is held prisoner in the elven city of Dietos, being captured on a mission to expose the underground passage of that very elven city.

Elf Lord: Danongale

Danongale is an elf, born to the house Diethis, keepers of the city Dietos. Due to his birth right Danongale has become keeper of the shrine swords in the city of Dietos. He is the rebirth of the elf granted the right to use the life essence so many years ago. Fated to battle Miotis again in this life, Danongale has been sent to the breaking.

Elf Warrior

These elf warriors are very agile and quick. Their knowledge of the forest allows them to use trees and vegetation for camoflauge purposes. They are a very worthy opponent and very persistent in defeating the adversaries. Loyal to Lord Danongale, these warriors will give up their lives for their kingdom.



Fellow Comrad

As Miotis journeys back to the human camp, he befriends fellow humans to help fight with him against the elf warriors. These fellow comrades are strong and will prove useful in supporting Miotis as he fights his way through the forest.

2.7 Levels

To complete “The Breaking Point” there will be three main zones that the player must move through. These are the Elven Fortress, the Forest, and the Breaking. The goal for the human character is to escape from the Fortress, make his way through the forest, and reach the border, where he will face the elven officer hero that is pursuing him. In the Forest, there will be a number of “side zones” which will have various outposts, skirmishes, and other groups of characters that will be playing out the ongoing war around the player. These side zones will also be where a majority of the Subquests are completed. These side zones will be detailed and designed in the next week. In addition, we will be adding more detail to the descriptions of the three main levels.

2.7.1 Elven Fortress

As a human character, after the game’s introduction, the player will find himself in a stone cell. The back wall of the cell will be blown open, and a human soldier will be standing in the opening. The wall will open into a large ceremony hall, where a number of unarmed elven priests and a small number of armed guards will be standing. At the player’s end of the hall is a stand with two blue-bladed swords. Taking up one of these swords, the player strikes down the guards, and escapes from the hall into the passageways. The passageways sprawl like a maze through the fortress, leading back into the cell blocks (which are especially well guarded, but sub quests may lead you through there), past dining rooms, council chambers, and eventually out to the civilian courtyard. There will be very limited NPC interaction in this zone, as most elves will either respond to the player with fear or anger, running away or attacking. The exception to this would be any other human prisoners (or spies) you may come across. After exploring in the courtyard, the player will find a merchant’s wagon where he can secret himself away and be transported through the main gate.

2.7.2 The Forest

When the forest level begins, the player finds himself outside the wagon in which he escaped the fortress, and three elven guards surround him. Either by defeating them, or running into the woods to buy some breathing room, the player continues his escape into the forest. The forest is a large, generally open area, consisting of a path much longer than it is wide. The player makes his way along the length of the forest, fighting elven patrols, battling off search parties, and helping any human NPCs he finds along the way. The forest, while the flow generally pushes along the length, is of a decent width, and there is no set “path” through the trees. The player can thus move a good deal from side to side as well, stumbling into clearings, or finding denser patches of trees. Along the edges of the zone, however, the trees are so densely packed that the player cannot pass them, which keeps the player moving generally in the direction of the border. There are certain places on the edges of the zone, though, where there are “openings”. These openings allow the player to move into different forest zones, which provide additional enemies and quests. As the player moves through the forest, he may come across other human soldiers that are also out in the battle. He can talk to these and may be able to get information about elven squad movements, the pursuing forces that the human reconnaissance has spotted, as well as possibly getting subquests, and just general back-story about the world and the war going on all around. This is where the majority of the game action will take place, and there will be a lot of combat and people moving through here. The ultimate goal of the player is to move all the way through the forest to try to reach The Breaking (the border), where the lines between the human and elven forces are strongest. When he reaches this point, he will be given the option to move on into the final zone.

2.7.3 The Breaking

The Breaking is the border where the regions of power between the elven forces and the human forces meet. Makeshift battlements will be laid along either side, but in the middle the player will meet very few other people, if he runs across any. That is, except for the pursuing forces. The elven hero and his squad approach the player, the other blue-bladed elven holy sword in hand, and you step forward into the center of the border zone. His comrades and the player’s both hang back, and, after the cutscene showing the approach is completed, the player must defeat him in single combat. If the player is triumphant, he has won the game.

2.8 Scripts

Scripts will be created soon.

2.9 Cut scenes

There will only be two major cut scenes in the game.

2.9.1 Battle Scene

The first cut scene will take place at the beginning of the game. This introductory cut scene will be of sketches and drawings depicting the battle between elves and humans and of the capture of the human. Narration will also accompany this brief cut scene.

2.9.2 Final fight Scene

The last major cut scene wilh be created if time permits. It will illustrate the human and the last elven antagonist preparing to fight each other gloriously, in the final combat scene.

2.10 Artificial Intelligence

The artificial intelligence included needs to focus on strategy. Placement of foes within areas of the game (ie. ambush encounters, etc.) alone can do a lot to promote the feel of thinking opponents in the war torn setting of our game. However, a smooth AI that makes opponents and allies act smoothly and strategically realistic is a necessity for this adventure.

2.10.1. Movement

2.10.1.1. Pathfinding AI

The first task is to set up a pathfinding system that will act the same for all moving objects with the exception of the player controlled hero. This will need to be implemented so that moving objects will avoid obstacles in a "line of sight" fashion. This way the object won't necessitate bumping into an obstacle before moving around it, ruining the experience.

2.10.1.2. Movement AI Influenced by Tactics

Depending on the tactics and loyalty of the object, movement choices made by an object can be influenced. For instance: if a support troop soldier for the hero has had his tactics set to Attack on Sight, the moment an enemy appears that soldier will choose to move to the vicinity of the first thing he sees. If a support troop archer has the same tactic, they will move into bow range instead of melee range upon sighting the enemy. Similarly a support troop who has been told to Hold Formation will continue movement within the formation, regardless of enemies sited.

2.10.2. Opponent AI

Opponents must have various combat AI that provides challenge to the player, so that the player must use different strategies to overcome those challenges.

2.10.2.1. Enemy Hero AI

The enemy hero is a special case; this opponent must be wily and difficult to beat. It will be a good idea to start with a base AI, and have some play testing performed against players while running a learning script so that the enemy hero appears especially cunning when the game is released due to its learning. Continuing with its learning during game play can provide additional challenges.

2.10.2.2. Opponent Minion AI

The enemy hero will have support troops similar to the player's hero has, and will act in much the same way, responding to the Enemy Hero's AI. All other opponent minions will act under the minion AI, which will be a basic tactical assignment. Again, initial positioning and grouping is a big factor.

2.10.3. Friendly NPC AI

Friendly NPCs will simply walk upon predesigned path cycles until approached by the hero. They will respond with violence only if attacked. (optional whether to allow this or make Friendly NPCs invulnerable).

2.10.4. Friendly Minion AI

The support troop of the player's hero need to respond to the player's tactical assignments as implemented in the movement section and actually initiating forms of combat.

2.10.5. Tactical Choices

These are the tactical choices available to the hero to give to support troops, and to be assigned to opponent NPCs:

Aggression:
Attack on Sight: Troops immediately attack foes in eyesight
Assist: Troops attack foes when they engage hero or troops
Non-combatant: Troops focus on movement and avoid fighting unless cornered

Order:
Break Formation: Don't hold formation during combat
Hold Formation: Hold formation during combat

Formation:
Point: Hero takes the forefront of a wedge
Wedge: Hero stays in the middle of wedge
Charge: Hero takes the forefront of line ranks of troops
Ranks: Hero stays behind lined ranks of troops

Line: Troops follow the hero in single file

3. Functional Requirements

The system will consist of a single computer that will perform the following functions.

3.1. Character Selection

The system must be able to allow the player to view and select from the characters available from play

3.2. Active Character

The client must be able to control the active character and view the environment surrounding the character from that character's perspective.

3.2.1. Rendering

The client must render the images that represent the background, objects, and effects of the zone the character is within.

3.2.2. Interface

The client must include an interface through which the user can control the active character.

3.2.3. Music

The client must be able to play appropriate music.

4. Environmental Requirements

4.1 Technical Specs

“The Breaking Point” Client will be able to run on any Windows operating system that supports DirectX of at least version 8.1. “The Breaking Point” requires a processor of at least a 400MHz, a Direct X compliant video card of at least 64MB RAM memory, at least 256MB RAM of memory, and at least 500MB of hard drive space (this is a safe estimate, it could be lower or higher depending on actual engine implementations). An optional requirement to enhance game play is a DirectX compliant soundcard.

“The Breaking Point” will be programmed under the Microsoft Visual Studio C++ programming environment. We will use Direct X for input, sound, and graphics. It is possible that some sample code will be used for the graphics engine. All code used will be cited.

The basic parts of the game will be the graphics engine and the game engine. These two portions are more theoretical than actual classes, and thus some functionality will overlap between the two.

The graphics engine will consist of textures, collision detection, and display cases. It will also consist of other functionalities of Direct X including audio and player input. Levels will be included in the overlap between the graphics engine and the game engine. For the purpose of level creation the graphics engine will be used, but for the purpose of placement in the story the game engine will have control. A portion of the player input and all of the character models will be encapsulated within a larger character class.

The character class will be a part of the game engine. The base character class will consist of statistics, weapon, equipped items, and magic skills. NPCs and Player classes will inherit from the Character class. The NPCs will have an embedded AI bot to determine which moves to choose. Player moves will be determined by user input. Players will also have an inventory, and an equipped skill set. The equipped skill set will determine which moves the character will use in response to user input.

In order to preserve our work we will meet every Friday to exchange our latest working code. This way if there ever is an error, the game will be in a working version as recent as one week’s worth of work. This will also reduce risk in that every programmer will have an updated version of all of the other’s code, so if one programmer loses work, only his or her week of work will be lost.

4.2 ADT for Graphics Engine

```
class vGraphicsEngine
{
  public:
        vGraphicsEngine();              // default constructor
        vGraphicsEngine(
                IDirect3DDevice8* curDevice);       // create the engine to use an already
                                        //  initialized D3D device

        ~vGraphicsEngine();

        bool startEngine(                       // returns TRUE if successful
                HWND theWindow,                 // the window to render in
                bool FullScreen,                // TRUE for full screen
                long width,                             // screen width (resolution)
                long height);                   // screen height

        IDirect3DDevice8* getDevice();          // return a reference to the display device

        bool setViewpoint(
                float fromX, float fromY, float fromZ,          // camera location
                float toX, float toY, float toZ);               // point to look-at (direction)


        bool addObject(
                vEngineObject* objectIn);       // add an object to the engine to be
                                                                // considered for
rendering


        vEngineObject* getObject(               // returns the object ref. or NULL
                char* name);                            // the name of the object


        bool setAmbientLight(
                float Red, float Green, float Blue);

        bool registerMesh(vMeshObject* aMesh);

        bool addLight(
                vLightSource lightIn);          // a light object containing the light info.


        bool set2DInterface(
                v2DInterface nonRenderedInterface);             // v2Dinterface is a class
```

```
                                                                                //
containing the info needed
                                                                                // to
draw a 2D interface.


        bool clearWorld();              // use this function to remove all objects and lights.


// call this function to draw all the objects that the graphics engine has been given
// (or the subset including visible objects when culling is completed)
        bool renderWorld(
                int curTime);   // the current frame (time); needed to see if  we need
                                                //to move through animation frames, or if
we've
                                                // "skipped frames.


  protected:
          IDirect3D8* d3dObject;
          IDirect3DDevice8* theDevice;
          GraphicObject* theObjects;
          GraphicObject* objectsTail;
          vView viewPoint;
          vMeshObject* theMeshes;
          float ambientLight[3];
          long screenWidth;
          long screenHeight;
};


// the vEngineObject class encapsulates all common data to game objects, whether
//  they're walls, trees, weapons, or even characters.  This includes all data relevant to
//  graphics and game state (mesh/model references, animation sets, world locations, etc.),
//  and also the functions to set/use this data.  To give an object more info, this class
//  will have derived classes (i.e. character, weapon, etc.)

class vEngineObject
{
 public:
        char* objectName;

        vEngineObject();

        vEngineObject(
                vGraphicsEngine* parent);               // reference to graphics engine.
        ~vEngineObject();

        char* getObjectType();              // each derived class returns the name of
                                                    //  itself, so the game engine
knows how
                                                    //  to treat each one.


        bool setName(char* nameIn);

        bool setLocation(
                float x, float y, float z);             // the location in the 2-D level plane


        bool loadMesh(
                char* fileName);            // file name of the .X file

        bool loadMesh(
                vMeshObject* meshIn);       // an already loaded mesh object.

        vMeshObject* getMesh();             // returns a reference to this object's mesh
                                                    //  object or NULL if no mesh
is set.


        bool startAnimation(
                char* animationName,        // the name of the animation to start
                int curTime,                // the current frame (so the animation knows
                                                    //  when it started, so it knows
which frame
                                                    //   to output when an update
is requested).
                float relXChange = 0,       // total x-distance to move by end of anim.
                float relYChange = 0,       // total y-distance to move by end of anim.
                float multiplier = 1);      // the relative speed of the animation (1/2
                                                            // to run in half the
time, 1.5 to run 50%
                                                            // faster

        bool  moveObject(
                float xChange, float yChange);          // the change from the current
                                                                    // position
(pos. for forward,
                                                                    // neg. for
backward)
```

```
        bool rotateObject(
                float xRotate, float yRotate, float zRotate);   // degrees to rotate from

                // current facing

        bool setRotation(
                float xFacing,                    // set the facing of the object in the
                float yFacing,                    //  world; default is aligned perp.
                float zFacing);                   //  to the xy-plane, facing along the
                                                          //  y-axis

        bool setScale(
                float relLRWidth,            // the relative left-right width (x-axis scale)
                float relFBWidth,            // the relative front-back width (y-axis)
                float relHeight);            // the relative height (z-axis scale)


        bool doAnimation(
                int curTime);                // the current frame, to be compared to the
                                             //  start time saved in the object to calculate
                                             //  the current animation frame.

        bool drawObject();                   // transform (update) and render the object

    protected:
        vGraphicsEngine* renderEngine;
        float location[3];
        vMeshObject* theMesh;
        char* curAnim;
        bool inAnim;
        int animStart;
        int lastAnim;
        int animEnd;
        float animXChange;
        float animYChange;
        float animMult;
        float xRotation;                // in radians
        float yRotation;                //   "  "
        float zRotation;                //   "  "
        float xScale;
        float yScale;
        float zScale;

};
```

```
class vMeshObject
{
 public:
        LPD3DXMESH stanMesh;
        ID3DXSkinMesh* skinnedMesh;
        D3DMATERIAL8* materialList;
        DWORD numMats;
        char* meshName;
        vMeshObject* next;
        IDirect3DTexture8** textureList;

        vMeshObject();
        ~vMeshObject();
        bool loadMesh(char* fileName, vGraphicsEngine* renderer);
};



typedef struct GraphicObject
{
  vEngineObject* theObject;
  GraphicObject* next;

  GraphicObject()
  {
        theObject = NULL;
        next = NULL;
  }

  GraphicObject(vEngineObject* objectIn, GraphicObject* listIn)
  {
        theObject = objectIn;
        next = listIn;
  }

  ~GraphicObject()
  {
        if (theObject != NULL)
              delete theObject;
  }

} GraphicObject;
```

```
// the vLevel structure contains all the information about a level.  This just gets the idea
//  of what a level consists of; it may be added to/subtracted from/modified throughout
//  development.

struct vLevel
{
        vLevel();                          // default constructor, sets all fields to NULL/0

        vLevel(char* fileName);            // loads a level from the file fileName.

        vEngineObject* levelObjects;            // An array of all the objects in the level
                                                //  (including trees, players, items, etc.,
                                                //  all which are instances or derived from
                                                //  the vEngineObject class.

        int numObjects;                         // size of the above array/list.

        vLightSource* levelLights               // An array of all the lights in the level.
        int ambientRed;                         // the level's ambient color.
        int ambientGreen;
        int ambientBlue;
}
```

5. Software Qualities

5.1 User-friendliness

User-friendless is an important quality for “The Breaking Point”. A user’s inability to understand how to use “The Breaking Point” can result in user dissatisfaction.

5.2 Correctness

Correctness is an important quality for “The Breaking Point”. If the game does not correctly model this document, user dissatisfaction can occur.

5.3 Reliability

Reliability is an important quality for “The Breaking Point”. If the game is not reliable, it can result in user dissatisfaction.

5.4 Performance

Performance is an important quality for “The Breaking Point”. If the game does not match up to the required performance, user dissatisfaction can occur.

5.5 Reusability

Reusability is not an important quality for “The Breaking Point”. Users will only have a single opportunity to view and test the system.

5.6 Extensibility

Extensibility is not an important quality for “The Breaking Point”. Users will only have a single opportunity to view and test the system.

5.7 Evolvability

Evolvability is not an important quality for “The Breaking Point”. Users will only have a single opportunity to view and test the system.

5.8 Robustness

Robustness is an important quality for “The Breaking Point”. Errors can result in loss of user control, which can result in user dissatisfaction.

5.9 Verifiability

Verifiability is an important quality for “The Breaking Point”. The system must be able to be tested throughout, in order to insure correctness throughout.

5.10 Maintainability

Maintainability is not an important quality for “The Breaking Point”. Users will only have a single opportunity to view and test the system.

5.11 Repairability

Repairability is an important quality for “The Breaking Point”. The system needs to be implemented in such a way that the administrative team can easily correct errors.

5.12 Safety

Safety is not an important quality for “The Breaking Point”. This system is for entertainment purposes, and should not pose to be a safety issue to its users or administrative team.

5.13 Portability

Portability is not an important quality for “The Breaking Point”. This system is for use with Windows™ operating system and DirectX™ visualization system, and portability constraints are contained within those two background systems.

5.14 Understandability

Understandability is an important quality for “The Breaking Point”. If understandability is an issue, professors looking over the code will become annoyed.

5.15 Interoperability

Interoperability is not an important quality for “The Breaking Point”. The system should only be for use with the operating system and visualization system of the user, and is not for use with other third party software.

5.16 Productivity

Productivity is not an important quality for “The Breaking Point“. No revenue will be generated by this project.

5.17 Size

Size is not an important quality for “The Breaking Point“. The system should easily fit on any computer designed to run it.

5.18 Timeliness

Timeliness is an important quality for “The Breaking Point“. If “The Breaking Point” is not implemented on time, professors will become annoyed.

5.19 Visibility

Visibility is not an important quality for <MMORPG>. The system will not be available to the public.

6. Time Schedule and Personnel

Design Document Due Date: 4/24/2003
Rendering Engine Creation: 5/09/03

Prototype Implementation Date: 5/28/03
Implementation Due Date: 6/10/2003

6.1 Schedule

Deadline Date	Goal
5th week: Friday May 2, 2003	• Storyline completion
6th week: Friday May 9, 2003	• Working on the game/graphics engine (basic mesh and input, basic rules for the game engine)
7th week: Friday May 16, 2003	• Work on the character class • Basic combat action such as sword • Attack list • Statistics • Character models (main), • Some basic AI (a very playable version) • Culling
8th week: Friday, May 23, 2003	• Main level (prison, forest map, some basic scripts, objects placed) • Leveling up system • Minimal AI battle • Basic enemy attacks • Collision attacks • Basic magic system implemented • Dialogue of game
9th week: Friday, May 30, 2003	• Enemies attack completed – AI • Separated AI script for the opposing hero • Subquests given by possible comrades within prison/town zones • Magic completed archery system implementation
10th week: Friday, June 6, 2003	• Henchmen/comrade AI • Forests subquests completion • Cut scene for the final ending • Sound effects and music
Finals week: Tuesday, June 10, 2003	• Completed Delivery

In accomplishing our intended goals per week, the designers of the game will first accomplish the human campaign. Due to time constraints, the parallel elven campaign will be optional throughout the design process

6.2 Deliverables by team members

Title	Team Member	Responsibility
Project Manager for Graphics Engine	Mike Bartlett	Responsible for seeking an existing graphics engine or to design a feasible graphics engine
Project Manager for Game Engine	Gary Garon	Responsible for seeking an existing game engine or to design a feasible game engine
Project Manager for the Art	Cathy Goings	In charge of creating the models of the characters as well as the necessary artwork that will be needed for the game (drawing sketches, design)
Project Manager for AI	Matthew Peronto	Accountable for overlooking the AI that is involved in the game.

6.3 Additional Research

Team Member	Research
Mike Bartlett	The majority of what I need to learn will be how to effectively use DirectX, using Direct3D, DirectSound, and DirectInput to manage the interface (including graphics, sound, and input). In order to make the whole graphics engine and make it run efficiently, I'll probably need to learn some additional techniques for culling and collision detection.
Gary Garon	
Cathy Goings	I will need to refresh some skills in the art of modeling, especially using the software 3D Studio Max. I am also willing to learn Rhino 3D. In helping out with the programming aspect, I will also need to learn DirectX.
Matthew Peronto	I need to learn about the large game system integration and graphic rendering and animation.

7. Potential Risks

If “The Breaking Point” is not released in a timely fashion or has quality issues, user dissatisfaction and professor annoyance can occur, which would bring a loss to the grades of the members of Visions. The only type of risk the system can partake of is the risk to the grades of Visions, due to the nature of the system being for entertainment purposes.

8. Glossary

Brush: Objects in a game zone that do not interact with players except to be visually displayed and be immovable aspects of the zone environment.

Zone: A portion of the <Adventure> game world which includes background brushes and game objects. Each Zone is designed for the player to move through it while completing his quests and progressing through the game. Also known as a Level.

Texture: The images which cover brushes, imported as image files.

Object: A game object within a zone that a player can interact with. This includes traps, NPCs, players, and other interactive environment features (destructible walls, trees, etc).

Character: A specific type of Object that refers to a person (human, elf, or other) or animal that has a set of statistics and skills defining its abilities and that can be interacted with through dialogue or combat.

Player Character (or Hero): The character that is directly controlled by the player. Also, one of the two main characters in the game's story.

NPC: Non-playing character.

Comrade: A character that follows the Player Character, and can be given limited orders.

Weapons: A type of object used by Characters to make Attacks.

Armor: A type of object worn by Characters to protect them from Attacks.

Inventory: A collection of Weapon and Armor objects that are carried by the Player Character.

Statistics: The three main attributes of a character, describing their physical power, ability to move quickly and precisely, and their magical power (Power, Agility, and Spirit)

Determined Statistics: Additional statistics that describe the characteristics of a character using the Statistics, Equipment, and Skills of that character. These include Life, Protection Rating, etc.

Attack: An interaction function, which causes a character to initiate a combat maneuver, usually on another game character.

Secondary Attack: An interaction function which causes a character to use a magical attack skill, usually on another game character, or on himself in some cases.

Use: An interaction function which allows the user to perform various actions with the zone, including but not limited to opening door objects, talking to NPCs, or using zone transfer points.

Dialogue – Having a conversation with NPCs in the game, involving hearing their speech lines, and responding with one of a selection of response lines.

Dodge – A maneuver that can be made by a character in order to avoid an Attack.

Parry – A maneuver that can be made by a character using a weapon or shield to block an Attack.

Counter-Attack – A maneuver that can be made against an attacking character when their attack is Parried.

Magical-Attack – An attack using a magical attack type skill that uses Energy with each use to create a magical effect to do damage to another character.

Quest: A goal or a set of goals given to the player to be completed in the current zone in order to progress through the game or retrieve a reward.

Subquest: A specific type of Quest that is not necessary for progression through the game, but that can provide extra story, gameplay, and/or rewards.

Rendering: The process of displaying models and brushes on a screen via the video card.

Animation: The model that is linked with a game object or the particle effect linked with an effect.

Skill: The different abilities and maneuvers a character can use when in combat to give them bonuses to their attacks and defenses.

Skill Type: The type of action a skill is used to undertake. The types are Attack, Dodge, Parry, and Counter-Attack, and magical attack.

Skill-Set: A set of skills including one skill from each (melee) skill type that will be used when in combat.

Skill-Use: A value associated with a skill that increases with the successful use of that skill.

Milestone: A point at which a Player Character gains new Statistics and Skills, reached by making a certain number of kills.

Race: The “teams” a player has a choice of when deciding which to start on, elf or human.

Model: A model that is rendered by the graphics engine and eventually the video card, associated with an object.

Particle Effect: Special effects rendered by the video card, associated with a skill or effect.

9. References

Due to the educational nature of this system, all the ideas of this document and this system are part of public usage.

9.1 Literature

"Programming Role Playing Games with DirectX" by Jim Adams

9.2 Games

We would also like to note the inspirational value of previous game systems such as Zelda 64, Final Fantasy, Dragon Warrior, Warcraft, Everquest, Dark Age of Camelot, and Shadowbane, and Advanced Dungeons and Dragons as well as various fantasy novels and mythology upon the mindset of the development team.