Prince of Pride: Social AI in Games

A Major Qualifying Project Report:

Submitted to the faculty

of the

Worcester Polytechnic Institute

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

by

_____

Francis Collins

_____

Ross Doran

_____

Gregory Kinneman

_____

Ryan LaSante

Date: March 5, 2009

Approved:

_____

Professor Robert W. Lindeman, Advisor

_____

Professor George D.J. Phillies, Advisor

Abstract

for the design of
Prince of Pride: A Social AI Game
By
Francis Collins
Ross Doran
Greg Kinneman
Ryan LaSante

This is the Final Report for the Major Qualifying Project *Prince of Pride: Social AI in Games*. The project consists of the planning, design, development, and completion of a fully playable video game showcasing the use of a middleware artificial intelligence (AI) system. This game is directed toward children ages 8-12 as an educational tool. It focuses on developing problem-solving and teamwork skills.

This document discusses the entire development of the project from the original Final Design Review onward. This document is comprised of the story, gameplay, art and technological aspects of the game *Prince of Pride* along with the social AI middleware. The game is a single-player, third-person, two-dimensional game with a cartoon art style. The main focus of the game is the balancing and management of emotions within a pride of lions and dealing with the different situations that can arise through interaction with a social AI. The middleware focuses on the social interactions of small groups.

Acknowledgements

The *Prince of Pride*, Social AI in Games MQP development team would like to acknowledge our advisors, Rob Lindeman and George Phillies. In particular, we want to thank Professor Phillies for suggesting his books to us. We would also like to thank all our testers for helping us to smooth out our game. We would like to extend our gratitude to the African Percussion Ensemble at WPI for allowing us to use their music in our game. Finally, we would like to thank Jon Marokhovsky who was forced to leave our development team early in the process, but whose suggestions, ideas and contributions helped shape our project into what it is now.

Authorship

Table of Contents

List of Figures

List of Tables

List of Acronyms

AI – Artificial Intelligence

APE – African Percussion Ensemble

API – Application Programming Interface

FDR – Final Design Review

UI – User Interface

WPI – Worcester Polytechnic Institute

Executive Summary

Many videogames with social interactions have started to emerge onto the mainstream gaming scene. Games like The Sims and Dragon Age have all started to focus on the importance of these interactions (The Sims; BioWare). As unscripted conversations and social situations in games become more prominent, it is necessary to improve these interactions to a level that imitates normal human-to-human interaction sufficiently. It is through the science of artificial intelligence that this is accomplished.

We created a piece of middleware that will exhibit this social artificial intelligence and demonstrate it in a simple game. *Prince of Pride* is a game for children wherein the player manages the day-to-day lives of a pride of lions in order to maximize the pride's happiness.

This social AI framework is hidden from the player, but its presence is made visible through the interactions of lions in this game and their reactions to the player and each other. Through management of these interactions, the player is able to manipulate their outcome in a way that is beneficial to the pride. The game utilizes 2D sprite animation, sound effects and textual dialogue to communicate the concepts of the game to the player.

Prince of Pride is the culmination of these efforts. Through the casual interface of a child's game, the user can experience the social intelligence system in a fun and interactive way. Whether the player is a game developer looking to use the artificial intelligence middle-ware or simply a child, *Prince of Pride* delivers an entertaining showcase of the benefit of social intelligence in games.

1 Introduction

In coming up with an idea for a game, the development team had many ideas of what to create. The themes included bureaucratic corporate sabotage and shipwrecked pirates and the gameplay ranged from tactical shooter to city-building to story-heavy role-playing. Amidst this disorientation and indecision, one choice was clear, the group wanted to do a project to build an artificial intelligence (AI) to model social dynamics. One theme which was brought up, but didn't immediately catch on, was a story about lions crossing the African savannah. Yet the more the team members attempted to describe to each other their ideas of how the AI would work, the more that they would use the lions as an example. As the first few weeks of the project transpired, the group became more attached to the idea of a game about lions. By the third week, the team had come up with a name for their project, *Prince of Pride*.

1.1 One Sentence Description

Take control of a pride of lions, protect its members and prove yourself to be a great leader.

1.2 One Paragraph Description

As prince of a pride of lions, you must offset the shortcomings of your father's harsh, unforgiving rule. Can you strengthen the bonds of the pride through cooperation and coordination? Can you gain the respect of the youth and the elders through noble and brave acts? Will you stabilize the pride, help them survive in the savannah, and lead them to a better life?

1.3 The Final Report

This paper describes the process and decisions made by the development team in the creation of their game. Chapter 2 describes the story and cast of characters. Chapter 3 describes the mechanics and type of gameplay. Chapter 4 deals with music, sound and visual artistic aspects. Chapter 5 deals with the technical implementation of the game. Chapter 6 describes the social dynamics artificial intelligence engine. Chapter 7 explains the development process the team used. Chapter 8 explains the testing and revision system used by the team. Finally, Chapter 9 is a post-mortem summarizing why decisions were made or changed and how these impacted the final product.

2 Story

The structure for the game is based off the events that the story follows. The story takes place over the course of four days. At the beginning of the game, the player is presented with the initial setting of the game through textual feedback. The player then has the day to solve the problem presented to the player on that day. It then turns to night, where the player receives feedback on how they did with regard to the story event as well as information regarding the event for the following day. This happens four times over four days and nights and then the player is finally judged based on how they were able to handle those events.

2.1 Story Overview

Prince of Pride is about a pride of lions that live in the African savannah. The player controls the lion, Sefu, prince of the pride. The pride is about to enter some trying times in which a series of events will occur leaving the members of the pride worried and saddened. It is the job of the player to do all they can to help the pride through these desperate times. The emotional state and well being of the pride depend on how well the player is able to handle these events. Along with the help of the other members of the pride, aside from his mean and cruel father Kojo, Sefu will do all he can to make the pride as happy as possible. At the conclusion of the game, the player will be awarded a score based on how well they handled the unfortunate events that occurred throughout the game.

This is a game about team and resource management that is aimed toward children ages 8-12. It teaches them how to manage a group, not just when everything is going well, but when there is hardship and adversity involved. It teaches the children not only to keep their composure, but to go through the proper steps as member of a group to make sure everyone else is keeping their composure as well.

2.1.1 Characters

Table 1 gives a brief description of each member of the pride and how they are associated with the other members of the pride.

Table 1 Character Descriptions

Character	Description
Kojo	Kojo is king of the pride. He is a harsh and unforgiving ruler. His dark mane and fur not only symbolize his malice, but his bitterness in his old age.
Sefu	Sefu is prince of the pride and a stark contrast to his father. He is a gentle and caring lion, though his strength is matched by few on the plains. He is intelligent and positive, and his light complexion radiates the comfort and safety he brings to those around his.
Amadi	Amadi is the warrior of the pride. Though old and weakening, he still maintains the mental strength and fortitude that he had in his youth. He, like Sefu, considers himself a protector of the pride. He cares for all, even Kojo, the one who cares for no one.
Zabia	Zabia is the motherly figure of the pride. She provides the warmth and kindness that everyone else in the pride needs. She is Binta's mother and has a strong emotional connection to her daughter.
Binta	Binta is the cub of the pride. She is full of energy and exuberance. Whenever a member of the pride is feeling down, she is the one to boost their morale with her unshakeable positivity. Her light fur emits the passion she has in making other people feel better.

2.1.2 Job Description

Each member of the pride has a specific job that they do to try and keep the pride together and as happy as possible or vice versa. The player is already doing all that they can to keep the pride's emotional state high, but the other members of the pride will also do what they can to contribute. The affects of their jobs, and the associated actions, can have either a positive or negative effect on the pride. Table 2 below explains the jobs of the different members of the pride.

Table 2 Job Descriptions

Name	Job	Description	Associated Actions
Kojo	Ruler	Since Kojo is a selfish ruler, he goes around roaring at the other members of the pride to make them feel worse, which in turn, makes him feel better.	Roaring
Sefu	Stabilizer	Sefu is trying to keep the pride together, and since he is the only lion capable of picking up items in the game, he is in charge of taking items from one place or lion in the game and giving it to another lion or place in the game.	Taking/Giving
Amadi	Warrior	Amadi is responsible for keeping everyone healthy and strong. He trains with the other members of the pride every day. Since swiping is so important to the hunt, that is what he practices with them.	Swiping
Zabia	Mother	Zabia cares for the other members of the pride, making them feel better. She does this by jumping. She used to use other methods of comforting, but after seeing the effect her daughter's jumping had on everyone, she was inspired by her daughter's actions.	Jumping
Binta	Booster	Binta is so full of positive energy that she can't contain herself. She lets off this steam by jumping around and entertaining everyone. She uses this to transfer her positive energy to the rest of the pride.	Jumping

2.1.3 Needs Description

The general need of the pride is to keep the other members of the pride as happy as possible. However, the other needs of the pride members are dependent on which stage of the story the game is currently in. Table 3 is a table describing those needs in relation to the story of the game.

Table 3 Needs Description

Story Event	Needs
Meat Giving	Each member of the pride, aside from Kojo, is very hungry and needs some food.
Binta Missing	The other members of the pride are very worried about losing Binta, except Kojo. They need you to find her.
Medicine Giving	Zabia becomes very sick and needs your help to get the medicine she needs to get better.
Skull Giving	The other members of the pride need you to present Kojo with the skull as a gift and change the fortunes of the pride.

2.1.4 Story Events

There is four story events that the player proceeds through during the course of the game. These story events will be the main method of gameplay for the player. It will be the player's job to investigate the pride, find out exactly what they want, which directly relates to the story, and do it. These story events act as the structure for the game with the first story event starting the game and the last one concluding the game. They run consecutively, so the next story event in the sequence starts immediately after the previous one has concluded.

The story events do not continue playing out until the player has figured out what they have to do, but will eventually conclude with the end of the day. There will be feedback presented to the player during the nighttime transition to explain to the player whether they were successful in their task or not. The next day will then present a new story event with the inability to return to a previously unsuccessful story event. For instance, if Binta is not found, that will have a dramatic effect on the player's final score at the end of the game. Table 4 explains the intricacies of each story event and Figure 1explains the transitioning through the story events over the course of the game.

Table 4 Story Event Descriptions

Story Event	Description
Meat Giving	The hunt from the previous night was a big success and there is a lot of meat to be eaten. However, since Kojo is a mean ruler, he is going to take all the meat for himself. Everyone else is very hungry and desperately in need of some food. It is up to Sefu to sneak the meat away from Kojo to the other lions and make sure they are nourished.
Binta Missing	Over the course of the second night, Binta has somehow gone missing. Everyone is searching for her, but she cannot be found. Following the hints from the other lions, Sefu must find where Binta has gone to. The hints will be presented in incremental steps until the other lions are close to telling Sefu where she is. Then the rest is up to Sefu.
Medicine Giving	It seems that Zabia may have eaten something bad and has gotten sick. However, as luck would have it, Amadi has some extra herbs. Sefu must get these herbs from Amadi to Zabia before she becomes too sick to help.
Skull Giving	After Kojo begins to see how much the rest of the pride likes you because of your kindness, he begins to open up to you about why he is so mean. The rest of the pride then realizes they just need to show how much they appreciate him to get him to change his ways. Binta found an antelope skull and it is Sefu's job to present that skull to Kojo as a gift to help the pride.

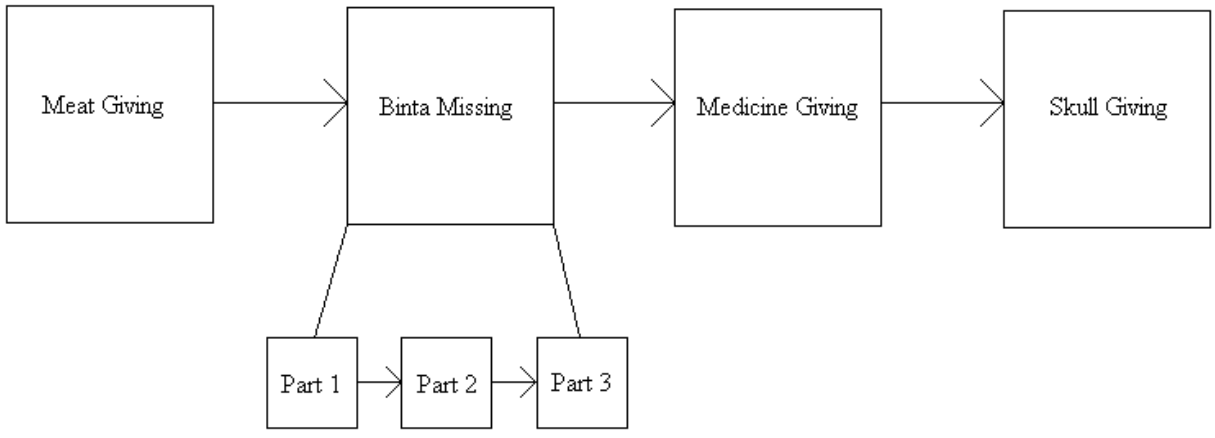


Figure 1 Story Event Progression

2.1.5 The Map

The setting for this game is the African savannah. The pride's territorial reign is only over a portion of the savannah, so they keep to their section. The pride will be interacting in the savannah environment shown in Figure 2.

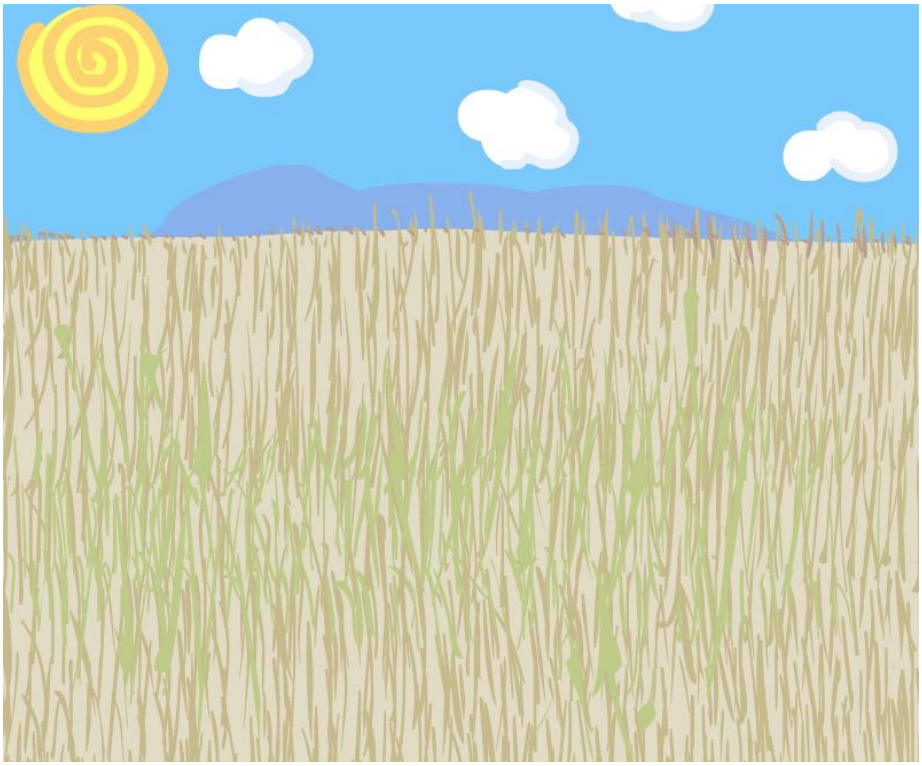


Figure 2 Savannah Map

2.1.6 Emotions

The emotional status of the members of the pride will be represented with an integer that ranges in value from 0 (very sad) to 100 (very happy). Below is a physical representation of this emotional concept (Figure 3).

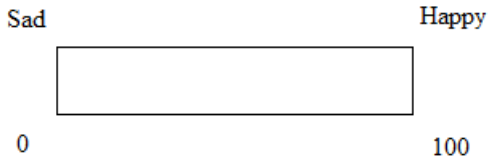


Figure 3 Emotion Bar

Each pride member's happiness will be set to an initial value of zero because the status of the pride when the player arrives on the scene is rather depressing. All of the different actions in the game will have different effects on the lions. The player's actions will have a greater impact than those of the other members. Below is a table that shows the effects of all the different actions in the game (Table 5).

Table 5 Action Effects

Action	Taken By	Effect
Talk	Sefu	+1
Take	Sefu	0
Give	Sefu	+25
Look*	Sefu	+25
Look	Sefu	0
Roar	Kojo	-5
Swipe	Amadi	+5
Jump	Binta	+5
Jump	Zabia	+5

* Special Look related to finding Binta

Talk is the only auxiliary action. It is not necessary to the story of the game. It is a representation of the fact that if the player were to complete the story event in the allotted time and had some free time, the player could do auxiliary actions on other lions to get ahead on trying to make the pride as happy as possible. Also, the player can receive positive or negative feedback from a member of the pride even if the story event has not yet been completed, but feedback during the nighttime transition will reflect whether the story event has been successfully completed or not.

2.1.7 Screenplay

The dialogue of the different members of the pride is dependent on the part of the story the game is currently in. The opening sequence of the game describes to the player what the game is about and what kind of a lion each member of the pride is. In game, both the textual dialogue and the actions that the lions take are reflective of the type of lion each one of them is. See Appendix A for the script of the game.

2.2 Original Story

2.2.1 Overview

Prince of Pride was originally intended to be an extensive game. The plot for the game was that the player takes on the role of Sefu, the prince of a pride of lions. The pride was leaving the grazing plains during the dry season in hopes of reaching an oasis, their only hope for survival. However, this trip would not only be hindered by the harsh elements of the African plains, but also by the harsh rule of Sefu's father, Kojo. It was the player's job to work with the other members of the pride to survive the trying journey to the oasis. The player had to combat the strict hand of his father and keep the pride together.

This was a resource and team management based game that was aimed toward teenagers in middle school and early high school. It was going to teach about the importance of awareness, communication, and the advantages and disadvantages of working in a team.

2.2.2 Characters

Making this journey was a pride of ten lions, including the player. There was also going to be emotional connections between lions. For instance, if the cub of the pride was feeling bad, the mother of the cub would, in turn, feel just as bad if not more so. All other aspects have remained consistent from the original to the final version of the game.

2.2.3 Story Events

We had a basis for a story event developed that dealt with trading goods and food with other members of the pride, but this was the most underdeveloped part of the story. However, there were also going to be cinema during the game at critical points. The critical points in our game were the beginning, when Sefu could have potentially been exiled, and the conclusion of the game.

2.2.4 Map

The game was originally designed for multiple levels, where the pride would rest for a couple days and then proceed to the next stop on their journey during the night. There were going to be a total of five levels, each with different features that were going to play into the story of the game, as well as the team and resource management of the player.

2.2.5 Hunting

Hunting was going to be the feedback to the player after each day of pride management. Depending on how well the player was able to manage the other lions and their emotions, the pride was either going to be fruitful in their hunting efforts or be left to struggle even more from lack of food the next day. This constant feedback and emotional state of the pride would

continue to be reinforced to make it clear to the player whether they had to change their methods of management or not.

2.2.6 Emotions

The emotions of the pride members were determined by two main elements, the passage of time and the resourcefulness of the player. The overall well being of the pride was determined by how many beneficial actions the player could do toward the other members of the pride in the amount of time the player was given. The social artificial intelligence (social AI) of the game was minimally involved in the story and outcome of the game.

2.2.7 Difficulties

There were also going to be multiple levels of difficulty introduced to the game so that the social AI would be more apparent if the player was playing the game on an easier difficulty. The lions that the social AI controlled would assist more in caring for the other members of the pride the easier the game was, and the emotional state of the pride would be more dependent on the player the harder the game got.

Unfortunately, a lot of this story development had to be cut due to unforeseen, and in some cases seen but not fully realized consequences. The following section goes into more detail about the reasoning behind making the changes that were made from the original to the final version of the game. For more details as to the original structure of the story, please refer to Appendix B (Original Final Design Review [FDR]). The story description is Section 2 of the original FDR.

2.3 Original to Final: Story

As is apparent from comparing the original story of the game to the final version, a lot of the content was cut. However, this should not be viewed as an entirely negative circumstance. A lot of what was cut was cut due to unforeseen circumstance that any group trying to complete a large project under time constraints must realize. There were extenuating circumstances that doubled the effects of this natural process and made things seem much worse than they actually became. Despite it all, the project still came together.

Among the features of the story cut were the number of characters, levels, and difficulties. Though this added variety would have been a nice addition to the game, the other characters and levels were not essential to the main elements of our game that we were trying to get across to the player. The difficulties would only have made the game more accessible as a learning tool to more children. This would have been a very nice feature, but not a core component. With the loss of an artist and a programmer, cuts had to be made, and these auxiliary features of the game seemed to make the most sense to go first.

Hunting was an aspect of the game meant to provide the player with feedback as the game progressed, so that they knew how they were doing, and would know to change things if they were not fairing as well as they had hoped. Though this had to be cut due to time constraints, we did not abandon the principle entirely. We decided upon a much more efficient, but just as effective story element of the nighttime transition. It gave the player the feedback they needed without requiring the man hours that the other feature would have.

The nighttime transition in-between each day of gameplay relays to the player how they had faired for that day. It does not give detailed feedback about each of the characters and how they felt as the day concluded, but instead gives a general summary. Depending on whether the player was successful in completing that day's story events or not, the nighttime transitioning slide would either relay positive or negative textual feedback to the player about their pride management for the previous day. They can then use this feedback to alter their play the next day. It has the same effect as the hunting, but was a much more efficient use of development time.

The one element of the game that was expanded from the original story to the final was the detailed development of story events. Story events were somewhat of an afterthought when first developing the story of the game. This was a serious flaw in our original design development. With the introduction of these detailed story events to the game, there is a structure to the game. The player has an overall goal for the game introduced at the beginning, as well as goals for each day of the game. This makes for much more motivated and involved gameplay than there would be had there just been an open environment for the player to move around in.

At the end of the game the players receive a score indicating how well they did. This game follows a structure shows the player they are making progress, how well they are doing, and then analyzing their overall experience in the form of the final score.

3 Gameplay

3.1 Events

The story events are what drive the gameplay and overall setting of *Prince of Pride*. Every distinct change in the setting or situation of the game is the result of a story event. The opening sequence of the game is a series of slides that brief the player as to exactly what they are getting themselves into with this game. The controls, setting, background, and character descriptions are all explained to the player in the form of a narrative at the beginning of the game. The story then segues from the narrative to the portion of the story that runs simultaneously with the gameplay. As the game begins, so does the first of four story events.

3.1.1 Meat Giving

The first story event requires Sefu to take the meat that was captured during the hunt the previous night and disburse it to the members of the pride before Kojo eats it all. Three pieces of meat are all that remain the next morning, and Kojo plans on eating the remaining meat and allowing everyone else to go hungry. Sefu can take the pieces of meat and give them to the members of the pride. Kojo already has a full belly so he does not need any extra food. When a member of the pride receives a piece of meat, the player will receive positive feedback in the form of the receiver of the meat having their happiness increased by 25 for each piece of meat they receive. Also, if the player talks to the lion after giving them the meat, the lion thanks the player for helping them get some food. However, if the player does not strategize correctly, and does not get all the meat from Kojo to the other members of the pride before the day is complete, the more general feedback during the nighttime transition will not provide positive feedback indicating the successful completion of the story event's goal. However, if the player takes all of the meat and gives it away then Kojo becomes upset, making the player's choice tougher than it first appears.

3.1.2 Binta Missing

The second story event involves the cub, Binta, going missing during the night. Aside from Kojo, the rest of the pride is very worried and is asking for the player's help in finding her. This story event is split into three parts. In the first section the information that the other members of the pride provide in assisting the player to find her is minimal. They are just worried, but really don't know where to look yet.

As the story event proceeds into the next section, Kojo gets more irritated with you if you interrogate him about the whereabouts of the cub, but Amadi provides you with some very vital information that helps you narrow your search. In the first part of the story event, the whole savannah is open for investigation. However, after speaking to Amadi in the second iteration of the story event, the player is able to narrow the search to the upper half of the savannah.

In the last part of the story event, Amadi gives the player the final clue in finding the missing cub. Since the layout of the savannah only has one bush in the upper half, and Amadi tells the player that he saw something in the bushes, the only plausible place for Binta to have wandered off to was behind the bush. However, if at any point during this story event, the player happens to stumble upon Binta behind the bush, the other iterations of the story event will not play out because they are no longer helpful to the player.

3.1.3 Medicine Giving

The third story event is when Zabia gets sick. Caring for her cub and being forced to hunt every night she becomes worn down and becomes sick. Amadi is in possession of some medicine that is the cure for Zabia's sickness, but he is unable to administer it to her because he is an older lion with unsteady hands. This is where Sefu comes in. It is up to him to get the medicine from Amadi to Zabia before it is too late. Seeing as this is a children's game, Zabia will not die from the sickness. In fact, she will get better for the next day regardless of whether the player successfully completes the story event or not. However, it will be apparent to the player that they did nothing to help the situation. This lack of performance will be apparent from the negative feedback on the nighttime transition screen and the damage it will do to the player's final score.

3.1.4 Skull Giving

The final story event is the most important and ties into the main focus and goal of the game. The player was introduced to a weary pride of unhappy lions, but up to this point they have just been working to keep things afloat. This is the opportunity during the game to change the fortune of the pride. This is the story event that really makes the player feel like they accomplished an important goal while being a member of this pride and a sense of completion that is crucial to the game experience. In the final story event Binta found a skull of an antelope and is keeping it as a toy. Amadi comes up with the idea to use the skull as a gift to Kojo to make him happy again. Kojo also reinforces this idea by expressing a want to only be appreciated and then he would be nice again. Binta eventually comes to terms with the player taking her toy, and if the skull is given to Kojo before the conclusion of the game, the positive feedback from the players, the final nighttime transition, and the final score will express the success the player has been attempting at all game.

3.1.5 Story Event Algorithm

As seen in the structure of the story events presented above, there is one algorithm used when creating the story events in this game. The player first enters the day with little or no knowledge, depending on how much information is expressed in the transition slides, of what the player is supposed to do that day. This forces the player to explore. Once they have explored the area, talked to the other members of the pride, and looked at the different items in the savannah, the player is now equipped with the information they need to begin solving the story event for that day.

If the player is an efficient investigator, they may even solve the story event before the allotted time has expired for that day. This is where the player can plan ahead. The main goal of the game is to keep the members of the pride as happy as possible. So the player can perform other auxiliary actions that are not dependent on the story, to supplement whatever help they gave to the lions during the story event. Unfortunately, due to time constraints, the only auxiliary action in *Prince of Pride* is talking to the other lions. However, this still simulates the idea that the player can utilize their free time with supplementary strategy. This idea is also enforced by the fact that the story events run consecutively. So if the player doesn't take advantage of any extra time they may have, they may not acquire the feedback from the game that they were hoping for.

Lastly, the story of the game must have a profound impact on the gameplay experience. If the story is just an extra feature, and not the main feature, the player might miss the entire point of the game. That is why the story's feedback is important in helping the player realize that they must do all that the story is guiding them to do or it will decrease the pleasure of the experience. This feedback is threefold. The dialogue feedback from the other lions, the textual feedback during the nighttime transition, and the final score all indicate to the player how important following the aspects of the story are to the gaming experience.

3.2 Actions

In *Prince of Pride* the way that characters interact with one another and the game world is through actions that they perform. Every item and lion in the game has a list of actions that can be performed on it, and each lion has a list of actions they can perform. Associated with each action that can be done to a character is an effect tag that tells what happens to the doer and the receiver of the action. This allows characters to easily figure out what actions bring them closest to achieving their personal needs.

All actions are set up in the same fashion. The Action class has a method, doAction, which, once called, lets the action class take care of changing the state of the game. The action tells the character if it needs to move and where, then once the lion is close enough it starts the action and at the end of the action performs any post action effects to the world. Inside the action class there is a class that handles performing the different actions. For each unique action there is one of these doAction classes. The doAction class keeps track of whether the action can still be performed and makes the actual changes to the game world in its startAction and doAction functions.

3.2.1 Player Actions

The player in *Prince of Pride* has a number of actions that they can use to interact with their surrounding environment shown in Table 6. There are two types of actions the player can perform. One is actions that change the state of the world, and the other type is finding out information about the objects in the game environment.

Table 6 Player Actions

Name	Description	Story Event
Give	This action allows Sefu to take an item from itself and give it to another lion.	
Look	This is the basic look action that Sefu uses when doing a general investigation of the map.	
Look	This look is specific to the Binta Missing story event. It is what allows Sefu to find Binta behind the bush.	Binta Missing
Look	This look is specific to the skull giving story event. It is what allows Sefu to find the skull behind the tree.	Skull Giving
Take	This action allows Sefu to take an item and remove from the world and place it within itself.	Meat Giving, Medicine Giving
Talk	This is the auxiliary action for the game. Sefu can use to help him investigate a story event or just to talk to another lion outside the context of the story.	

There are two different actions that the player can do to influence the game: Taking and giving. The Player can use the take action to either pick up an item or take an item from a character, such as picking up the meat or taking the medicine from Amadi in the game. By taking the items, some of the characters can become upset, for example when the player picks up all the meat Kojo gets upset since there is no longer any meat for him. The give action also changes characters based on the situation. For example if the player gives Zabia the medicine, Zabia then becomes much happier and more active. Through different uses of the give and take actions the player influences all of the other pride members. The look action is another action that is used in the game to change the state of the game. At certain points in the game, looking at objects will find hidden characters or items.

However, the look action is also used for information gathering. Looking, along with talking, at or with game objects allows the player become comfortable with their current objective and the current state of the game. When the player talks to lions, he/she get the opinions of the lions and can use these comments to get a better understanding of how the player's actions will affect the lion they are talking to. When they look at a lion, they get a general feeling about the current state of the lion, and possibly hints about what they need to increase their happiness. By talking and looking the player can get a better grasp of what they can do to most effectively increase the overall pride status.

3.2.2 AI Actions

While the player has these four actions that they can perform, the non-player characters also have their own actions which they can perform (Table 7). These actions each have a different effect on the statuses of the other lions and influence the overall state of the pride. For

example, Kojo can roar at anyone but Sefu, and when he does he gets happier. However, when Kojo roars at someone, the person being roared at becomes unhappier. Conversely, when Binta jumps at someone, it makes them happier while making her tired decreasing her overall happiness. These actions add variety and make it so that even without the player the pride will still maintain some sense of social interactions, but don't work to make the entire pride happy. They only serve to fulfill the needs of the individual lions, whatever they may be. It is only through the player's actions that the entire pride can reach its fullest potential.

Table 7 AI Actions

Lion	Action Name	Description
Binta	Jump	This action allows Binta to jump at the other lions.
Kojo	Roar	This action allows Kojo to roar at the other lions.
Amadi	Swipe	This action allows Amadi to swipe at the other lions.
Zabia	Jump	This action allows Zabia to jump at the other lions.
All	Nest	This action allows the lions to rest in-between actions. This is the no-op action. In a dynamic world, this is probably the most important action for the AI.

Throughout the beginning phase of the project, the team was planning on including a large number of actions to be available to the AI driven characters. However, as the project was scaled down the number of actions were one area that suffered. This is in part due to the fact that half of the lions were cut thus removing the need to have as many actions. Another reason was to limit the amount of work required from the artist. With every new action there had to be a new animation drawn in four different directions at least once, possibly as many as three times.

4 Artistic

4.1 Sound

Sound is extremely important to a game, especially to a game where many things are going on at the same time. It is difficult for the average game player to keep track of all the things they must do over the course of a game, but if a social AI is working on its own and the players are children, then audio feedback is critical to making sure the player knows exactly what is going on during the game. That is why a sound is associated with each action that the lions can perform aside from Sefu, who the player is watching visually most, if not all of the time. The player of the game cannot consistently monitor the actions of the other members of the pride while the player is also carrying out their duties, so sound is the player's means of measuring activity in the pride. Table 8 is a table describing the sounds associated with each lion and their corresponding actions, along with a brief description of why each type of sound was chosen.

Table 8 Lion Sounds

Lion	Action	Sound	Explanation
Kojo	Roar	Lion Roar	This is self explanatory. If the actual sound trying to be represented can be found, that's what is used. This is a realistic lion roar obtained from recording an actual lion roaring.
Amadi	Swipe	Swipe	This is a cartoon effect swipe. Since the game is designated toward children, the cartoon swipe is more deliberate in getting across what the lion is doing than a realistic sounding swipe.
Zabia	Jump	Boing	This is also a cartoon effect. There were more subtle jumping sounds available, but the message that the cartoon sounds are able to get across to children were more important than providing realistic lion jumping sounds. In fact, cartoon effects are specifically designed to stimulate the perception of a child, so they were the perfect choice.
Binta	Jump	Boing	(Same explanation as Zabia)

4.2 Visuals

Figure 4: The art in *Prince of Pride*

The team decided on the artistic style for *Prince of Pride* very early on in the design process. During the initial planning stage of the project, the team decided a two-dimensional (2D) cartoon style would fit the demographic and casual nature of the game the best, and it fit the sprite artist's previous experience. The team decided that for the lions to be relatable, they would be displayed in a "side-to-side" 2D view as shown in Figure 4 (as opposed to the Real Time Strategy top-down 2D view), and that the lions would move in four different directions.

The artist created asset lists in an Excel spreadsheet and used these lists to create preliminary sketches of each object. The artist then created outlines of these sketches, and then colored on a layer underneath the outlines to finish off the asset.

While the background, user interface (UI) and object assets were relatively simple to create, the lions proved to be the most challenging and time-intensive part of the art creation process. Due to the sheer number of animation frames, much time was spent applying changes to these frames and hand-outlining each one.

4.2.1 Background

The background is an important aspect of the game. It provides an atmospheric context for the player, and it allows both the player and the lions to be immersed into the environment created by the game. Background assets were created as separate images, which were later strategically placed in the game world based on the needs of the game as seen in Figure 5.

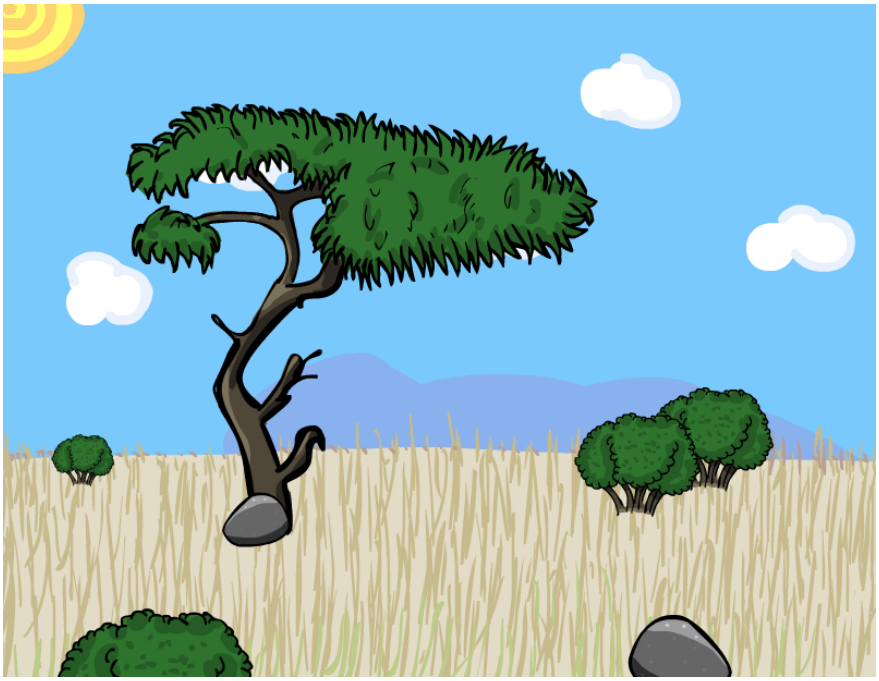


Figure 5: An early example configuration of background elements.

4.2.1.1 Plan

After the team determined that the game would be set in Africa, the artist made a list of background elements native to this environment such as rocks, trees and bushes, and placed this list into an Excel spreadsheet to form the background asset list (See **Appendix H**).

4.2.1.2 Implementation

Figure 6: An acacia tree reference image
(http://www.veeriku.tartu.ee/~ppensa/Acacia1.jpg)

Using some reference imagery such as Figure 6, the artist sketched the basic background elements in Adobe Flash CS4. Afterwards, the artist outlined, colored, and shaded the elements in Flash. Finally, the artist applied transparent backgrounds to the assets in Photoshop and scaled down to the appropriate size for the game.

4.2.1.3 Decisions Affecting Artwork

The only design decision that affected the creation of the background assets was the loss of the background artist. For more information about this, see our Post-Mortem.

4.2.2 Objects

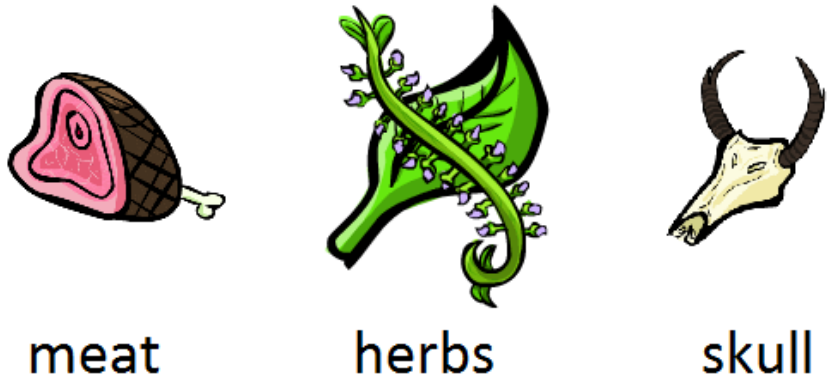


Figure 7: Game objects

Although the focus of the game was on the lions, some of the interactions of the lions required the presence of physical objects. Each story event relies on at least one game object, as shown in Figure 7.

4.2.2.1 Plan

From the initial game development discussions, it became clear that the game objects were going to be limited to only what was essential for the gameplay stages. The team determined these gameplay stages in the development phase, and decided that there would only be three game objects: Herbal Medicine, Meat, and an Antelope Skull.

4.2.2.2 Implementation

The process for creating the object assets was identical to the background assets. They were sketched, outlined, colored, highlighted, and shaded in Adobe Flash. Then, they were scaled and made transparent in Adobe Photoshop.

4.2.2.3 Decisions Affecting Artwork

The team did not develop an initial asset list during the design phase, because it was not entirely clear what the gameplay stages would entail. However, as soon as the gameplay stages were finalized the artist created the assets.

4.2.3 Lions

The lions proved to be the most complicated assets to complete. Each lion character had to have specific actions animated for them, and each action involved drawing and re-drawing at least 30 frames, because they had to be shown from 4 directions (the left and right sprites could be reversed).

4.2.3.1 Plan

During the design phase, the artist did some rudimentary research on youtube.com (http://youtube.com/results?search_query=lions%20zoo) and flickr.com (http://www.flickr.com/search/?q=lion&w=all) to find examples of lion movement. One example that proved particularly useful was this animated GIF, Figure 8, of a lion walking, from a website which is now defunct (http://www.3d-animated-gifs.blogspot.com/2008/11/lion-walking.html).

Figure 8: A frame from an animated walking lion image

The artist used these references to create conceptual drawings for the lion sprites, and submitted these for the team's approval. The team then decided all of the actions that lions would be performing in the game. These actions were used to create an asset list that detailed the four views of every single frame of each lion character's action animation (Appendix I). The artist trimmed down the asset list by making a variety of shortcuts. The lions were classified into three groups: Adult Males, Adult Females, and Male/Female Cubs. In this manner, the list could be separated into three lion types rather than the ten characters it had before.

Finally, the artist removed all animations where the lions faced left from the asset list, as they could be mirrored later. This cut down the list by 25% to a much more manageable size.

4.2.3.2 Implementation

The artist used the asset list to create preliminary sketches for each potential action animation. These sketches evoked the basic sense of movement that was required for each action, which helped the artist fix what looked wrong later.

Figure 9: These rough sketches provided the framework for the final lion animations

The artist then used these sketches to outline each of the sprites as seen in Figure 9. Using Adobe Flash, the artist created background layers containing the concept art, and drew on top of them to create fleshed-out, fully-inked black outlines of every lion action animation frame shown in Figure 10.

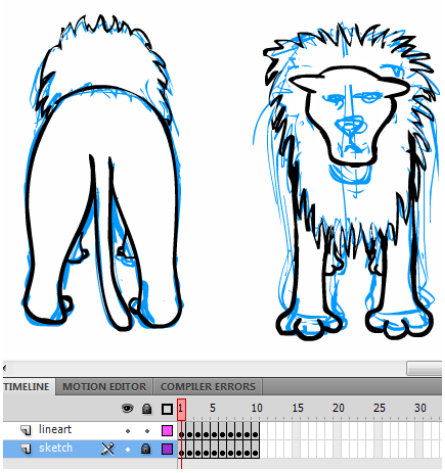


Figure 10: The lineart was drawn on top of the sketch layer.

To complete the outlines, the artist then animated the heads of each lion. The artist expedited this arduous process by creating a "Symbol" that represented each lion type's head, for each of the lion classification groups. Then, the artist rotated and moved the head, to represent the lion's head movement in each frame of the symbol. Finally, the artist could quickly switch the symbol frame into any of the ten characters. By doing this, the artist was able to animate the head movement just once for each action, and efficiently export animated lion bodies for every single lion character.

Finally, the lions were brought into Adobe Photoshop to be colored. Because there were so many frames for animation, the artist created an Adobe Photoshop macro to cut down on

some of the time used to color each lion. First, the artist used the magnetic lasso tool to select the entirety of the lion's body. Then, he filled the lion's body with a flat color. The artist painted the mane of the lion with a different color, and then ran the macro to place each lion in the center of a transparent background, and export the image in a standardized size for the game.

Additional frontal face images were created by the artist for later use in the user interface, by adapting the concept art and lion sprites to create emotive indications of a lion's current emotions.

4.2.3.3 Decisions Affecting Artwork

Many decisions affected the art asset creation process. Due to an unexpected personnel shift, the team lost a programmer and an artist (See Post-Mortem). Because of this setback, the team decided to halve the number of lion characters in the game, which allowed the sprite artist to work on the background assets and develop a user interface.

The scale of the lions led to a bug in the game relating to collision detection, so the assets of every lion had to be scaled down to accommodate this. The artist accomplished this with an Adobe Photoshop batch script.

The artist also made some decisions concerning the assets. Due to the time restraint, and loss of a partner, the artist decided not to include shading and highlighting on the lion sprites, which lessened the workload.

Another problem arose because of Adobe Flash. Because Adobe Flash can't use transparency, the artist had to take screenshots of each sprite in Flash, import them into Adobe Photoshop, and apply a macro that removed the white background of the image. This slowed down the process quite a bit, and in the future, the artist will ensure that the initial art program supports transparent backgrounds.

Finally, the artist's auto-center macro disrupted the flow of some of the animations, and many frames had to be re-positioned. Instead, the artist should have aligned the lion's feet to the bottom of the image, and made any further adjustments by hand.

4.2.4 User Interface

4.2.4.1 Plan

During the design phase of the game, the team discussed various types of user interfaces. Based on design considerations such as the casual nature of the game, and the requirement of seeing updated statistics at all time, the team decided that a Real-Time-Strategy point-and-click status-bar style would be the most effective for our game.

The artist created a paper version of this user interface, and the team tested the interface in-house by manipulating the paper elements. The artist also made a Shockwave Flash version of the interface, which was distributed internally and externally and approved by the testers.

Finally, the artist created placeholder User Interface boxes, to provide basic functionality to the alpha/beta versions of the game until the final user interface could be substituted in its place.

4.2.4.2 Implementation

The artist used dimensions provided from the game code to create an Adobe Photoshop guide layer that consisted of guide lines for each UI element. Then, the artist drew a preliminary sketch of the interface on the layer on top of these guides.

Then, the artist drew the outlines of the "branches" on a separate layer, and colored the interface on a layer below that (Figure 11). The artist also highlighted and shaded the branches.

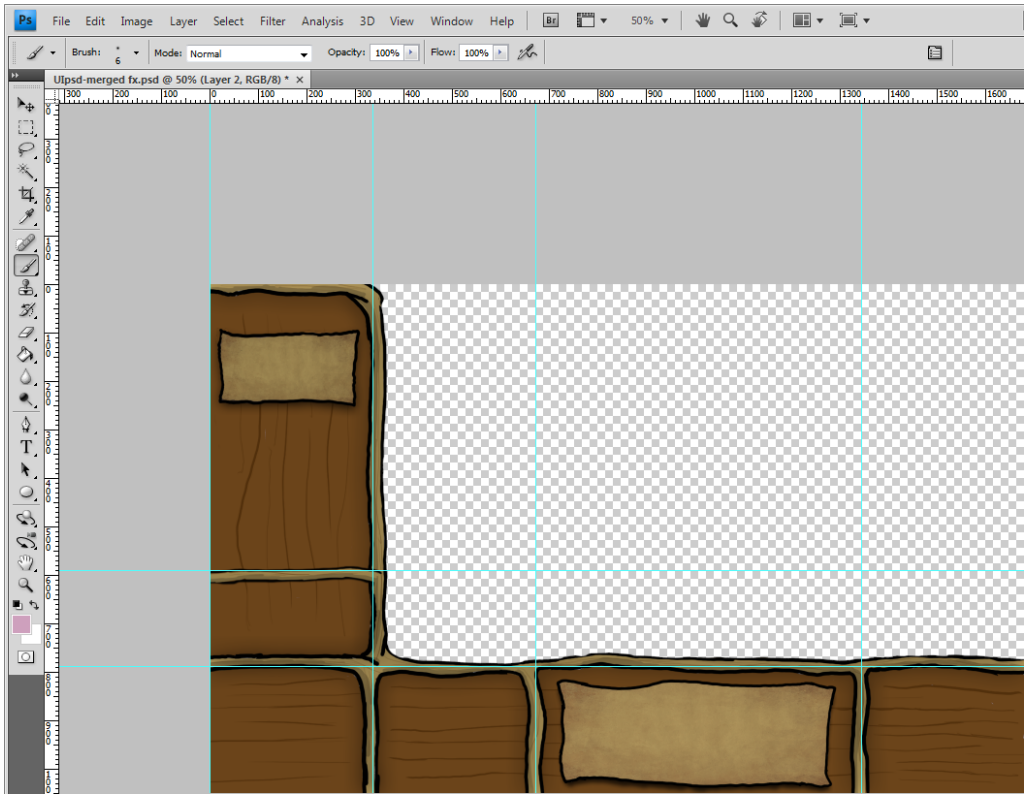


Figure 11: The User Interface in Adobe Photoshop

Finally, the artist added rudimentary layer effects in order to add aesthetic appeal. These effects included a drop shadow on the branches of the interface, as well as details like texturing the wood paneling and the paper message boxes.

4.2.4.3 Decisions AffectingArtwork

Due to the loss of the UI artist, the sprite artist had to create the interface and this led to cutbacks. For more details, see the Post-Mortem.

4.3 Music

The source and integration of music into our game arose from a personal contact of one of the team members who had several contacts in both the WPI African Percussion Ensemble (APE) and the Berkeley College of Music Video Game Music Club (VGMC). The APE has a set of recordings of African drum music which has been separated into different tracks by drum type. One goal that the team had was using the music in a dynamic way so that the music would serve as a form of player feedback. Several ideas were brought up as to the best method. The simplest idea was having volume increase as the player did better. A second one was having more drums come in as the player completed more story events. A feedback system more related to the pride status was also considered. This would involve either assigning each character a drum that only played when the character was happy, or a system where the overall pride status determined which drums were being played.

Further, in an attempt to add variety to the music, the VGMC would remix the samples from the APE to create a longer and more diverse soundtrack. As the focus of the project was on the WPI students and not their collaborators, these concepts were dropped from the final project. However, the game still features music from the APE as its soundtrack.

5 Technology

5.1 Game Engine

The team used the XNA Framework as a foundation for the construction of an engine on which *Prince of Pride* was built. The game engine that was constructed consisted of multiple parts that handle non-gameplay specific topics such as animation, sounds, file loading, helpful functions, input handling, rendering, and the user interface. The different components that the team used to handle these topics were the Animation Manager, Audio Manager, File Loading Manager, HelperFunctions, Input Manager, Renderable Objects, UI Manager, and the UI Viewer.

5.1.1 Animation

To handle animations in *Prince of Pride* we decided to create an Animation Manager. The team determined that the best course of action would be to create a singleton that could easily be called from anywhere to handle playing animations. This allowed for a very simple interface to playing animations that requires only one line of code from the component making the call.

When implementing the manager it was decided to use sprite sheets for each animation. The sprite sheets used were taken from Microsoft's SampleSpriteSheet project and are dynamically created (Microsoft). This allows for only needing to load the files once, while not forcing our artist to manually create all of these sprite sheets. Also the interface for using Microsoft's sprite sheets is very simple and allows for easy traversal between frames making playing animations simple and easy to use.

The Animation manager keeps a list of all the objects that are being animated and the information about their animation such as the start time, the number of frames, the current frame, the frame rate, the total play time of the animation, and whether the animation is looping. On every update the Animation Manager checks all of the objects to see if they need to go to the next frame. If the objects haven't started or a sufficient amount of time has not passed then the Animation Manager skips the object. Otherwise the manager will go to the next frame and if the object is at the end of the animation and it is looping then the Animation Manager will go to the beginning of the animation, otherwise it will remove the object from the list of objects to animate.

The Animation Manager has five methods that can be used for animating objects. They are Start Animation, Stop Animation, Pause Animation, Resume Animation, and Is Playing Animation. Start Animation starts a new animation for the passed object. Stop Animation removes the object from the list of objects to be animated and completely stops the animation without being able to resume until start animation is called again. Pause and Resume Animation pause and resumes the playing of the animations respectively, while Is Playing Animation returns a Boolean specifying whether the passed object is currently being animated or not.

5.1.2 Audio

Similarly to the Animation Manager for animations, we chose to use an Audio Manager to handle the playing of all sound effects and music. For *Prince of Pride*, the team once again decided to go with the singleton design in order to make playing audio as simple as possible. When looking into the Audio Manager we came across example code from Microsoft that implemented an Audio Manager that performed all of the tasks which we were planning for the Audio Manager so we decided to use the Audio Manager from the Minjie example (Microsoft). This Audio Manager was implemented as a singleton with commands to play music or sound effects that are loaded into the game. By using this asset the team was able to save a lot of time and work associated with creating and debugging a component to play audio files.

5.1.3 File Loading

When designing the project, the team knew that they wanted to be able to load in maps from files, making the creation of maps very simple. However, we needed to devise a way to parse it in order to accomplish this. We did this by creating a File Loading Manager. The manager is a singleton allowing for simple use of its methods from anywhere in the program. This manager combined with a small additional project, Prince of Pride Data, allows for the loading of characters and maps.

The File Loading Manager works by allowing the information to be first loaded in from the given file through the use of the MapData and CharacterData classes in the Prince of Pride Data project and the content pipeline provided by XNA. The data gets read in from an XML file that is set up in the same fashion as the MapData or CharacterData class and gets turned into the respective object. The File Loading Manager then creates a map or character based on the properties given in the file and then returns the game's internal representation of a map or character.

5.1.4 Helper Functions

At the beginning of the project it was not apparent that there would be a need for a large collection of general purpose helper functions but as soon as the implementation portion of the project began the need was obvious. The team then decided to create a static class that would contain all of the commonly used functions that were not specific to any one portion of the project. This allowed for easy organization of functions and created a resource for us to perform simple actions without having to rewrite code.

In the HelperFunctions Class there are twelve different functions ranging from getting the current scaling of the game, the scale of the map to actual screen coordinates, converting world coordinates to local coordinates, calculating the depth of an object, and calculating line intersections (Buckland, 2005). These functions allow for cleaner methods in the rest of the code base and reduce redundancy.

5.1.5 Input

Handling input from the user is a very important aspect of any interactive form of media, since without being able to take input the user can no longer interact with the program. It was clear to the team from the beginning that we would need a good way to handle the user's input. To solve this dilemma, an Input Manager was created to take user input and give the data to the appropriate components.

The Input Manager is a singleton class that handles both the user's input and the control of the mouse cursor. It polls both the keyboard and the mouse moving the cursor so that it follows the current mouse position. On mouse clicks, it determines whether it is a left click or right click, and whether the click was on the game window or one of the UI components. Then, it passes that information on to the main game.

5.1.6 Rendering

The rendering of object and components on the screen is important in any program. In the design phase of the project, the team agreed upon using a Rendering Manager that would draw on the screen all of the parts of the game that needed to be visually represented. This design planned on using a subscriber pattern in order to subscribe objects to the manager when they needed to be drawn and on each draw loop, the Rendering Manager would draw all of the objects that subscribed to the manager.

However, this idea of a Rendering Manager was scrapped when we realized that the XNA game had a Drawable Game Component that when added to the list of components in the game would allow the object to draw itself. This was more desirable since it was already implemented and allowed us to explicitly state how to draw each component in its own draw method rather than inside the Rendering Manager. It was noticed, however, that most objects shared common variables and methods, such as position, depth, scale, a texture, the portion of the texture to draw, and point of origin. Also, we decided that it would be beneficial to have a basic drawable component that would not need any extra work besides some default values to draw itself, so the Renderable Object was created.

The Renderable Object handles a lot of the tedious work required to draw to the screen and add an object to the list of components. It contains all the variables stated above and also has a default draw method that can be overwritten by any class that extends it. The Renderable Object serves as the base class for almost all of our visual objects in *Prince of Pride* including our lions, items, and level objects along with the mouse cursor. Using the Renderable Object has made it easier to deal with problems of objects going off screen and being set to invisible than using the Rendering Manager because it does not require removing and adding objects to the list of subscribers based on whether we want them drawn or not.

5.1.7 User Interface

From the beginning of the project the User Interface was looked at as a small part of the game, one that would be easily implemented and tested letting the developers move on quickly to the next part of the game. It was determined that there should be a UI Manager and a UI Viewer. However, during the planning phase of the project the team overlooked the complexity of creating a UI. The UI is made up of many different components that needed to be built, carefully placed, and tested. This portion of the game engine took the longest and was also the biggest surprise to the team in terms of what needed to be done versus what was initially thought.

5.1.7.1 UI Manager

The UI Manager is a singleton class that handles all the computing for the User Interface. It changes the UI Viewer to match the needs of the game. Along with the computing, it serves as the only connection between the game and the UI. All changes to the UI that the game wants to make must first be processed by the UI Manager and then it changes the UI Viewer as it sees fit.

The UI Manager has methods that set the list of actions the UI can display, set the text in the message box, handles mouse clicks on the UI, select objects, and pick up items. When an object is selected, the UI Manager sets the profile image, displays the object's status if it is a lion, and shows all the actions the player can take on the selected object. When the UI Manager handles picking up items, it sets the image and name on the section of the UI for items currently held by the player.

5.1.7.2 UI Viewer

The UI viewer is the main container for the visual aspects of the User Interface. This was one of the more time-consuming parts of the game engine because of the time required to test everything and place it properly so that everything was set up the way our artist had designed it.

The UI Viewer has the different UI components that combined make up the whole of the UI. These are the Actions Bar, the Game Object Status Viewer, the Game Object Viewer, the Held Item Viewer, the Pride Status Viewer, the Textbox, and the Time of Day Viewer. The Actions bar is where the current actions the player can take are displayed. The Game Object Status Viewer is where the information regarding the selected object is shown if the object is a lion. The Game Object Viewer shows the name of the current selected object, its profile image and if the object is a lion, the current action the lion is performing. The Held Item Viewer shows the image and name of whatever item the player is currently holding. The Pride Status Viewer displays the overall status of the entire pride. The Textbox is where all the messages to the player are shown; this component also handles text wrapping. The Time of Day Viewer displays a sun traveling across the top of the screen indicating the time left in the current day and loops back to the beginning when it reaches the end of the day.

5.1.8 Special Effects

During the initial design phase of the project it was decided to have a visual effect that changes the brightness and saturation of the colors of the game depending on the overall pride status, acting as another means of feedback on the player's current score. The plan was to use an Effects Manager that would apply any graphical effects to the game. However, this was removed from the project as an extraneous component when we scaled the project down to a manageable size for our team.

5.2 Lions

5.2.1 Creation

The process of turning the data that represents the lion in the XML file into the lions that the player sees is a three part process, as represented by the figure below (Figure 12). First the data that represents all the initial information about the map, the gaming world, and all its contents is stored in an XML file. This XML file must then be parsed by factories. In the context of this project, a factory is a function that interprets raw data and creates an object from that data that the game can use. The two factories created for this game were the Item and Lion factories. These two factories take the XML file's data and parse out the items that are to be placed on the map, their features, and where they should be placed. The same occurs when loading in the lions. Once all of the information has been parsed and the items and lions have been created within the context of the game, they can be added to the game environment and the game is ready to be played. See the figure below for a visual representation of the creation process.

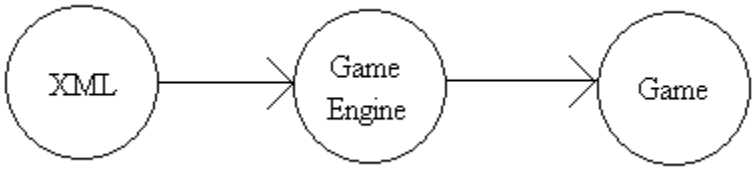


Figure 12: Lion Factory Process

5.2.2 Player-Controlled Lion

Through the use of the user interface the lion controlled by the player, Sefu, is able to pick up and take items and then give them to other lions. This is the main feature of the player's lion character. All the lions have a container within the class structure of the lion that allows them to carry an object of the game world, however the player is the only one with this feature active. This allows the player to remove an item from the map, store it in the lion, and then retrieve it at any time and perform an action with it (the give action). The user interface visually represents this software concept with a viewable held item box in the interface. This is synchronized and updates with the internal information on the lion's held item.

All lions have a lion controller associated with them that does the "thinking" of the lion. However, a lion controller, the base controller, branches off into two different types of controllers based on who is controlling the lion, either the player or the AI. This is shown in Figure 13. For the lion controlled by the player, its extension is a player controller. This controller deals with the gameplay mechanisms that are exclusive to the player, the actions that are exclusive to the player, and all the updates to the UI related to the player's section, actions, and held items.

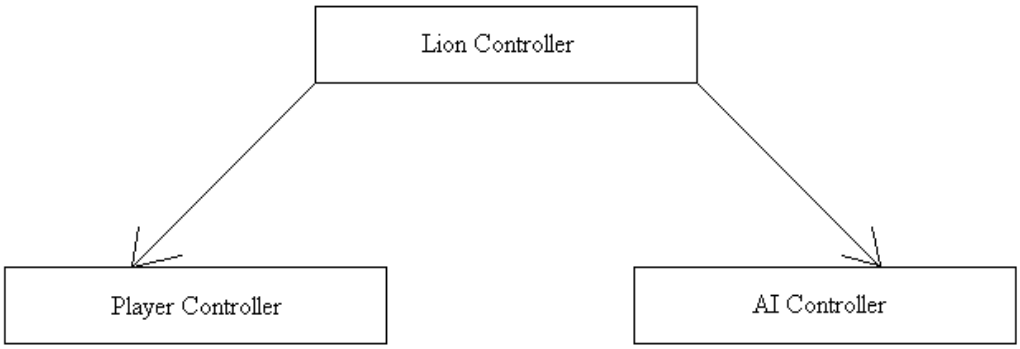


Figure 13: Controller Hiearchy

5.2.3 AI-Controlled Lion

As stated in the player controller section, an AI controller has the base lion controller. When it branches off into the AI controller, it gets features specific to the proper functioning of the AI. It retains all the basic functionality of the lion from the lion controller, but the lion's "thinking" is now dependent upon the AI of the game. Other than basic pathfinding, the main function of the game's AI is its communication with the social AI middleware. Each AI Controller has an associated character. This character uses the logic of the social AI to figure out how the AI controlled lion should conduct itself in the world. The specifics of these two different forms of AI, pathfinding and social AI, will be discussed in more detail later.

5.3 Events

5.3.1 Timeline

The timeline is the base for all time triggered events in the game. The timeline is composed of several alarms. Alarms are specific instances in time at which events are triggered. Each alarm contains a name for the alarm and the time when it is supposed to go off. The timeline is based off the XNA system time. When the game starts, the timeline initializes its start time to the time when the game started. All the times of the alarms in the timeline are relative to the time when the game was started. Regardless of when the game is started, the times for the alarms are still synchronized to go off at the appropriate times in the game. The names of the alarms correspond to a story event, and that story event has the same name so that name matching can be done when deciphering what story event should be triggered at the time the alarm goes off.

5.3.2 Story

The story is the superstructure containing all of the story events to be triggered throughout the game as well as in what order they are supposed to go off. The story events have a name that corresponds to the alarm that triggers them at the right time, all the information about the game that should be altered when that story event is triggered, and the story event that should next be triggered. When the game first starts, the story pauses the game cycle so that no gameplay is going while trying to introduce the player to the game. Once the opening sequence has finished, the game is resumed. The information about the first story event is initialized to the game as the game is resumed. The first story event then sets the time when the next story event should start. The general connect between story events is that a gameplay story event initializes when a nighttime transition slide should start. Then when the nighttime transition is running, which pauses the game, the morning story event is initialized. The morning story event will simply unpause the game and allow the initialization and running of the next gameplay story event. This cycle continues four times until the end of the game is reached.

6 Artificial Intelligence

6.1 Character Movement

For the character movement in *Prince of Pride*, the team initially planned on using pathfinding to allow for each character to easily calculate how to get to wherever they were going. However, pathfinding was replaced by steering behaviors as the project was scaled down. We came to this decision because we were able to lessen the amount of work while still being able to achieve comparable results. Since the number of lions was brought down from ten to only five, the chances of lions being unable to arrive at their destination due to obstructions was greatly reduced, meaning that moving directly towards the point that they wish to go to will be enough for the players to make it there. There is still a chance of a character getting stuck between objects and the edge of the map, but this problem is handled through level design.

There are three different steering behaviors that we implemented to help the characters move around the screen: Arrive, Avoid Objects, and Avoid Walls. The combination of these three behaviors makes for a believable sense of movement in a world space. All of three of these steering behaviors are modified versions of code from Mat Buckland's code examples from his book, *Programming Game AI by Example* (Buckland, 2005).

6.1.1 Arrive

The Arrive steering behavior is the basic component that allows a character to move from point A to point B. It allows for characters to accelerate to their maximum speed, move to the desired position, and then decelerate, stopping when they arrive at their destination. This is a relatively simple algorithm that will make sure that the character will move towards its destination and stop at it rather than going past the point, bouncing back and finally coming to a rest once it has slowed down enough to no longer overshoot the destination. The Arrive function works by taking in the destination where the character should arrive, and it then returns the amount of steering force needed to arrive at that spot while limiting itself to the maximum speed of the character (Buckland, 2005). Figure 14 depicts this arrive behavior.

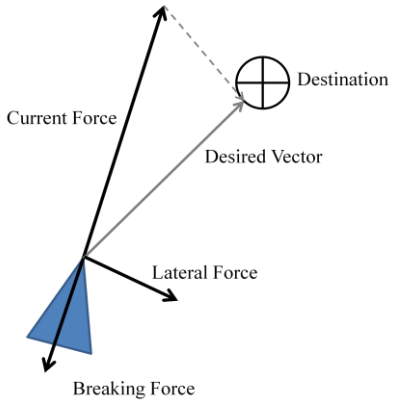


Figure 14: Arrive Behavior

6.1.2 Avoid Objects

Avoiding objects is an important aspect to the movements of characters. Without it the characters will travel through one another, which can create problems such as flickering when objects occupy the same space and breakage of immersion due to solid objects going through one another. Therefore, the Avoid Objects when combined with the Arrive steering behavior create a convincing interpretation of movement of solid characters in a populated world.

The Avoid Objects function works by first taking in the current velocity of the character. It then uses that data along with information regarding the position, size and mass of the character and creates a bubble around the current character to see what objects the character needs to avoid. It next calculates all of the objects that are intersecting the bubble and looks to see if it needs to avoid the object. After that, the algorithm then takes all of the objects it needs to avoid and takes the closest object and calculates the steering force required to avoid a collision (Buckland, 2005). Figure 15 shows how object avoidance works in *Prince of Pride*.

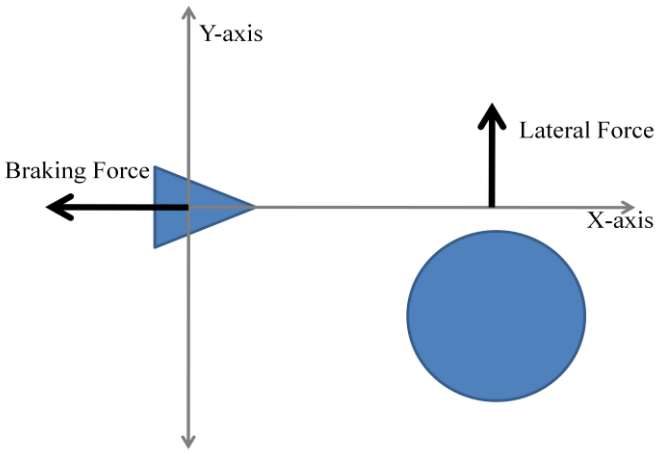


Figure 15: Avoid Objects Behavior

6.1.3 Avoid Walls

While avoiding objects is important, it is also important to keep the characters on the actual map. To do this, we added invisible walls around the borders of our maps. We then added the steering behavior of avoiding walls. The code to do this was taken from Buckland's book, and with some modifications, was able to make walls to keep the characters onscreen (Buckland, 2005).

Avoid Walls works by creating three feelers that extend out in front and to the side of the character. When one of these feelers runs into a wall, it then creates a force normal to the surface of the wall. It then returns the force pushing the character away from the wall where the closer the object is to the wall, the greater the force. This keeps all of the objects from going off the map as they are traversing the game world. Figure 16 shows how objects avoid walls in the game.

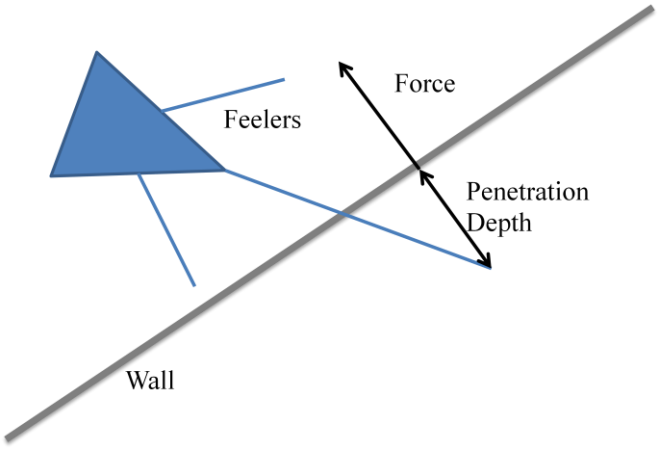


Figure 16: Avoid Walls Behavior

6.2 Social AI

As stated before, one of the first goals that the team agreed on was creating an artificial intelligence program to model social situations. We then decided that we wanted our AI to be isolated from the rest of our game so that it could be incorporated into any game in order to add social behaviors. To accomplish this, we first attempted to create a model of how individuals interact in certain situations. Upon coming up with a model, we implemented this model in C# so that it would be able to interface with our game. While developing the AI we realized that it would be necessary to have several programs to test the AI since we could not prove that the software was generalizable with only one game. Finally, we tied our games together with the AI to create social dynamics and relations.

6.2.1 Social Model

Our model of social interactions revolves around individuals that we call Characters. These individuals do not represent humans or lions, but rather decision-making entities. These entities have three primary attributes: Emotions, Attitudes and Needs. These attributes are used to determine all the interactions between Characters. Originally, there were also plans to include Social Groups, which would be groups of Characters connected by association. The final component is the AIManager which is responsible for maintaining and updating the state of the Characters.

Emotions reflect the emotional state of a Character. These emotions are represented by a number which indicates how strongly that emotion was being felt by the individual. In order to

allow the game designer flexibility, the balancing of Emotions and their values is intentionally vague. If a designer wants characters to be in binary states, then values of 0 and 1 could represent if emotions are being felt or not. Likewise, the emotion values could be represented as ranges, such as integers between 0 and 100, allowing a character to be, for example, angry, sad and tired all at once. Other possibilities that were considered included normalized values, where the sum of all the emotional values for a Character totaled 1.0, and interrelated values where some Emotion values were derived from the values of independent Emotions. A final type considered was opposing Emotions, such as individuals with depression who can appear happy sometimes but suddenly find themselves sad.

Attitudes are similar to Emotions, with the exception that instead of describing how a Character feels about itself it describes views towards other Characters. Where the Emotion 'anger' may allow a character to act in an angry manner towards everyone, the Attitude 'anger' would be specifically directed towards another Character. It is therefore possible to have individuals who either do not know other characters or are non-communicative with them, perhaps until introduced by a mutual friend. The exact choice of when and how to connect Characters was again left up to the game designer. An example of designer choice would be setting a limit of 10 close Characters with up to 3 Attitudes each (example: friendship, knowledge, jealousy) and a limit of 100 other Characters with only 1 Attitude (example: knowledge). Another choice would be conflicting Attitudes, such as love and hate, which could lead to strange, but realistic behavior, such as passive-aggressive tendencies.

The final attribute is Needs, which represent the goals of the Character. Often these are tied to emotions or attitudes, but this is not a requirement. While one goal may be to decrease tiredness below a threshold of 20 points, and another may be to have an average of 20 for the knowledge attitude towards the other characters, a need could also be something unrelated to attributes. This is because every need has a tag which associates the Need with a game object. The reason for this is to allow Characters to have Needs pertaining not to social behavior, but also to game-specific concepts. For example, an addict Character might have a need to smoke cigarettes, and while the importance is low and not connected to any Emotion, it will still affect how the Character behaves.

Originally there were two other attributes that we created for Characters that eventually became combined with Needs. These two were Importance and Actions. Importance reflected how vital certain Needs were to a Character, but these were eventually combined with Needs because we could order the needs and that gives us the order of importance. Actions were created so that games could call upon these Actions, and the Character would update its own attributes as well as those of other Characters. However, this was ineffective because Characters did not necessarily know about the state of other Characters, or even their existence, and therefore we felt that Actions would require changing our model significantly. Thus, the work of how to affect Characters was placed completely in game-space, away from the Social AI.

After Characters, we noticed that social behavior often revolves around class, race and social circle. As such, individuals may have conflicting views about any one Character. At first Social Groups were added to the model to make computations less complex, because Characters would interact with a single Group instead of multiple Characters. However, it became quickly apparent that this would not decrease complexity, and in fact, made things more complex. Further, it was realized that social groups do not behave as an average of all their members, but rather at the whims of their leaders. If the leader of one faction dislikes a character, then the Attitude of the gang itself will be negative, even if some of the faction members are friends with that character. Social Groups had Emotions and Attributes but not Needs, since groups lack the ability to perform actions (the actual action is the result of many individuals doing the same thing). However, during development it was found that Social Groups would not be useful to our game, since the final game had only five characters representing one group, and as such, this element was never integrated or fully developed.

The last component of our Social AI is the AI Manager. The manager handles access to the Characters in the AI and handles operations pertaining to all Characters, such as searching sorting and updating. The primary role of this manager was to make sure that the Characters would only be affected by the AI Manager, and access to the Characters was never directly given to the game. However, with Characters being so closely tied to game entities, this resulted in much slower access to information, so direct access to Characters was given to our game. The choice of whether to use the AI Manager or access Characters directly is up to the individual designer's decision. On one hand, access time in increased, while in the other there is a fixed entity through which the game interacts. The final responsibility, of updating and sorting Characters, is essential to making the Social AI work well with games. Whenever the game performs a loop, it simply needs to call the update() function present in the AI Manager.

6.2.2 Technology Demonstration

All combined, these components, created in C#, allowed us to have Characters which could act according to their Needs towards other Characters and themselves. However, as shown above, it became apparent that a single game would not be enough to show that our Social AI was versatile enough to work with several games. As such, we developed a technology demonstration that modeled the trading of items between people as they attempted to get their favorite foods. Characters that were not on friendly terms could not trade, and only meaningful trades, where at least one side benefited, and neither side lost, were allowed. This game, based around trading rather than preserving the health of a group, shows that our game can work for economic and non-economic situations alike.

6.2.3 Integration with the Game

As above, integrating our Social AI with our game, *Prince of Pride*, was rather easy to do. The most important consideration was how game individuals would interact with the Characters representing their personalities. After this, the next step was combining decision-making based on Needs with ways of achieving those Needs in the game. This was done by

combining factors such as distance to items and lions and the effectiveness of certain game actions with the importance and value of certain Needs. As such, Characters would be able to make optimal choices as to what would be the easiest task that would provide the most benefit to the Character.

The biggest problem with creating a Social AI which is independent of a game is that the AI can never actually make a decision; it can only modify the weight of certain decisions. As such, the game designer must understand how the AI works and build the game AI to take full advantage of the Social AI. As such, there is no way to loosely couple the Social AI and the game. This is different from a physics engine, which is able to create two models of the world which only interact in very small ways, most notably position and velocity. For example, most games with physics engines apply physics to inanimate objects which have no ability to control their movement. If these objects do have the ability to exert force, they are not usually controlled by an AI. Game characters are therefore only controlled by physics after being made inanimate, usually through their deaths. Further complicating the issue is that predicting even very simple social behavior is very difficult for psychologists and sociologists alike. By contrast, Newtonian kinematics are trivial. As such, the problem of creating a believable social AI is very complex, and by having some of the decision-making responsibility on the game designer, an isolated Social AI is unlikely to be very successful.

7 Methodology

When determining a general system for how to complete our project, our advisor suggested we break the work into three stages corresponding to the three terms we would be working on the project. These stages were design, implementation, and testing. He also suggested a waterfall method of planning out the work for the entire project before attempting to complete it.

Throughout development, the team attempted to follow the Gantt chart supported waterfall method, but instead ended up combining this with an iterative development system throughout the process. In this system, the team determined weekly what should be accomplished and the amount of work that the team was able to commit given other commitments. Deadlines imposed by the Gantt chart were loosened, providing flexibility as far as the order of some tasks, but forcing the team to occasionally take on more work during any given week. With greater flexibility, the team was able to adapt to events ranging from losing a team member to midterms to snow days. As such, the team was under a challenging, but not impossible, schedule every week.

8 Testing and Refining

As with any software project there was a need for testing all of the content we made. To do so we broke the testing up into two phases, internal testing and external testing. The internal testing was testing done by the team as a means of finding major game breaking bugs and any gameplay related problems. Then after internal testing of a component of the project, the team started to focus on external testing. This involved asking friends and family to play the game and look at the user interface and let us know what they thought and where there was room for improvement. By using both internal and external testing, the team was able to get a good sense of what worked with the project and what either needed to change or be taken out completely.

8.1 Internal Testing

The internal testing was the first round of testing on every single component of the project and was where most of the major issues were discovered. It was during the internal phase that the team was able to find many problems early on and fix them before they became any larger. This was generally easier than performing the external testing because as we went through the game or social AI, we were able to stop once we saw a problem and figure out how to fix it. Conversely when having someone else play test our game we had to let them play all the way through the game, hear their thoughts and then go about fixing problems they encountered.

8.1.1 User Interface

As stated in the Art visuals section above, we performed multiple tests for the user interface. The first being a paper model that the team artist created, where all the different components were pieces of paper and he walked through each step with the team in a sequence of events. This allowed the team to get a good sense of what it would look like generally and what the major components of the UI would be. During this phase some minor changes were applied but the overall general format was very similar to the final version found in *Prince of Pride*.

After the preliminary test for the UI, the artist then went on to make a Macromedia Flash demo of the new modified user interface. This was then shown to the team and went through a few more minor changes to come very close to the final version of the UI.

8.1.2 Game

The internal testing of the game started from the beginning of the implementation phase and has continued through to the very end of the project. In the beginning of the testing, problems were quickly found and easily changed, but as the game gained complexity bugs became more abstract and needed more thought and care to solve.

We began making a list of all the bugs that needed fixing and their level of concern. This allowed the team to prioritize what needed to be fixed versus what should be fixed only if there is enough time once all the more important problems were handled. Game breaking bugs took

first priority, and then followed by significant problems that broke immersion, such as animation flickering. Lastly came bugs that would provide polish but had implemented solutions that worked well enough for now. By implementing the bug tracking we were able to efficiently test and refine the game through the entire implementation and testing portions of the project.

8.1.3 Social AI

The team knew that it needed to have multiple ways to fully test all of the features that were integrated into the social AI. This lead to the creation of two tech demos that worked beside *Prince of Pride* providing different testing environments for the AI. The first instance of the social AI and its initial testing phase was using the first tech demo. In the tech demo there were two characters that had items the other wanted but would not trade with one another since they did not like each other, but they would trade with the player. So since the characters respected the player enough, they would trade with him, allowing for each character to get the item they wanted. This showed that the initial form of the AI worked and was where the first tests were performed.

After the first tech demo, the game became the main place that testing was performed. It was in the creation and observation of the game characters that showed problems with the design. While watching the characters interact without the player, we were able to identify where code could be optimized and what new information would be useful to pass on to the game. This was a much slower process than the previous testing because the team was watching the interactions take place in-game where actions took time to perform and were very hard to duplicate exact situations. When compared to a command line tech demo where everything was instantaneous and could create the same situation time and time again, the need for a new environment to test the latest version of the social AI.

This was when the team decided a second tech demo was necessary for testing the social AI along with providing another proof of concept for the AI middleware. By creating the new tech demo, the team was able to look more carefully at what was happening in the social AI without the game getting in the way. The second tech demo works by having two characters that have items the other wants, and then there are two other characters that can talk to either of the two parties. The two neutral characters act as mediators and allow for the two fighting parties to still trade their items to get the optimal item selection. This provided a more complex situation with autonomous actions allowing for a fuller testing environment than the first tech demo while still having the speed and simplicity lacking from the game.

8.2 External Testing

The majority of testing was done internally; however, the external testing was crucial in finding problems that were easily overlooked because of the team’s familiarity with the project.

8.2.1 User Interface

The user interface was externally tested in two different forms. First it was tested using the same Flash demo the team had used for internal testing of the UI. We showed some of our friends and our advisors the demo and received some critics on the placement of some of the aspects of the UI. These critics lead to the final design of the user interface, which would later be implemented into the game.

The UI was also tested through play tests of the game by friends and family. Throughout the tests the general consensus was that the UI worked well. It had all the information that was needed to accomplish the game. By letting others play without any information outside of that provided by the game, it became clear that players did not notice all of the features of the user interface until almost half way through the game. This is not that big of a problem since the game is only a couple of minutes long and it is natural that it should take some time to learn everything there is to see in the game. Also, it was not a factor because the testers were still able to complete the game successfully.

8.2.2 Game

Testing the game externally provided the team insights that were often overlooked by the team and helped to bring out the experience that we were attempting to make. We had our friends and family play the game while we watched. This let us notice what people missed, and when asked questions, some we answered while some we let them figure out things on their own. In general most things they figured out without any help. However, with any bugs that we noticed we answered any questions that they brought up about the problem.

By keeping our comments to ourselves, we were able to not only get their opinions, but also notice what bugs were being noticed and what ones that we knew of weren't being noticed. For example, a bug where the level's sun was still in the sky went unnoticed, but the bug that Kojo kept saying the same line after the player had already given away the skull item was immediately noticed.

The majority of information gained directly from the players' feedback was gameplay or bug related items, such as increasing movement speed and fixing message problems. Watching and talking to just one tester gave us more insight into what was needed to make *Prince of Pride* an entertaining game than all of our internal play testing combined. The bias that we have built up over the months of working on this project influenced our opinions about the game, but with the new perspectives provided by the testers we were able to improve the game and make it into an enjoyable experience.

9 Post-Mortem

In every project there are things that go right and things that go wrong. This postmortem is split up into three sections: the aspects of the project we thought were successful, the parts of the project that didn't go as planned, and what we would have done to improve upon the project as a whole. Each of these subsections discusses topics relating to the overall team, the artistic portion of the project, and the technology aspects.

9.1 What Went Right

Two of the biggest things that went right in the project was making a game that is playable and all of the assets that were created for this game. The game was successfully completed such that people can run it and play it from beginning to end; this in itself is a huge accomplishment given that we only had about twenty four weeks to design, implement and test the game. There was a considerable amount of artwork to be done by one artist and all of it looks great and works well in-game.

There are several different processes in the project that went correctly. The main obstacle that any team must overcome is adversity, and that presented itself in many forms to us over the course of the project. The first and most significant obstacle was the loss of a project team member. He was a double major, in both computer science and an IMGD major focused in art. When he left the team, we lost 1/3 of our computer science and ½ of our artistic capabilities for this project. Since we were unaware of this until after we had decided upon our final design, this required immediate cuts on both the artistic and artificial intelligence fronts. It also required a re-adjustment of the jobs that each of the group members was required to do for the project to come to completion. All of the art was now the sole responsibility of our other artist, and the programmers who were going to focus their efforts on the game engine, story, and gameplay, now had to take on the additional responsibility of assisting the lone programmer now in charge of putting together the social AI. These were the main reasons for the dramatic scaling that had to be done in the project; scaling that was unavoidable due to the circumstances. However, as a team we managed to adapt our project to fit the new team structure we had and still managed to produce a strong product.

The other main obstacle was the lack of a game. This aspect of the game has been a focus of much debate over the course of the project. Admittedly, there really was no game contained within the initial proposal of the project, but it has developed over the course of the project. Had this been dealt with at an earlier time, it most likely would have been better developed. However, even with the slow development of the game, a satisfactorily one was eventually created and incorporated into the game With all the different features of the game that had to be cut, this feature actually grew stronger and more evident. The player is given a problem, and must solve that problem over the course of the game. Learning from their environment, and the information

supplied to them, they must do all they can in the allotted time to obtain their reward and that is what our project has evolved into.

A few other decisions that we made ended up helping us to succeed in our efforts. We had daily meetings during the planning phase, and regular meetings during implementation to discuss what we had done and what we were planning to do before our next meeting. This allowed us to segment our project into very manageable pieces and we were able to accurately track our progress. We also chose a very good language to use for building our project. C# was a familiar language that allowed us to worry more about what were going to program rather than how we were going to program it. With good online help and good debugging tools, we were able to code efficiently and effectively.

As stated before, the loss of a team member and an underdeveloped gameplay process were the major setbacks to our progress. However, there was other adversity we had to face as well. Since our remaining artist was so swamped with the visual art of the game, he could not focus much effort on the audio of the game. It actually became another job of the technical team to produce the sounds that are present in the final version of the game. As such, the music and sounds are not highly varied and the audio feels somewhat hollow. However, the audio is functional, and seeing as the technical team had to adapt to art production, no one could really expect more than that.

There were many successes made by the art team as well. The "Symbol" idea for the lion heads saved the artist innumerable hours of animating head motion for every action, for each lion. The artist also developed many macros that helped cut down some of the work, by applying actions to over 300 frames at once, rather than having to do them individually. Finally, it is worth noting that although the UI/Background artist was unable to continue working on the project, the sprite artist successfully handled this extra workload without failing to deliver any of the assets required for the final version of the game.

Artistically, a lot of setbacks occurred simply because of the sheer number of frames there were for each lion. When any change had to be made, such as scaling the assets, applying transparent backgrounds, or centering each asset, the artist lost a lot of time applying the change to each of the 400 frames. In the future, it would behoove the artist to have a better plan for modifying the assets if the need arises.

9.2 What Went Wrong

While many things went right with our project, unfortunately, as with any project, there were things that went awry. In designing our project, we had made major efforts to stress the flexibility of our social AI. We planned two technical demos to accompany our game. However, as time went on, and schedules got tighter, the AI features we culled until only the technologies that would be used in Prince of Pride were developed. One of the technical demos was dropped

due to time constraints, and the concepts such as group identities were also lost due to the time constraints.

One area that reduced the effectiveness of the team was overspecialization. Roles were assigned at the end of A-term, and they remained that way for the rest of the project. This meant that some problems could only be solved by one team member. This forced some dependencies between the programmers. This caused some lulls in production when the services of a programmer were required and they were not able to address the problem in as timely a fashion as was necessary. The learning experience was slightly skewed by this too, and thus team members didn't receive as well rounded an understanding of the project as they would have had rotational programming been instated.

Early in development we were advised to look at where the game type and mechanic were being implemented. Unfortunately, this came up after the design phase, and the team was too focused on completing the implementation quickly rather than ensuring that the final game was fun and compelling. In some ways this was a failing of doing all the planning first. This issue that should have been caught and fixed during planning was instead allowed to slide indefinitely since the team was 'past planning' and it was inconvenient to change once noticed. As such, the gameplay is solving simple puzzles rather than any compelling strategy.

A final issue that the team was victim to, that most projects are susceptible to, was that our vision was greater than what we could achieve. While the team had lots of energy and time to spend on the project, we did not plan effectively considering that we had other classes and commitments to deal with. Even as we trimmed off excess from our game when we lost our team member, we still attempted more than was reasonable.

9.3 What We Would Change

In hindsight the team would approach the project differently in a couple of key ways. One thing the team would do differently is we would focus on the interactions of a small group in order to really show the social dynamics. By having a game built around the social dynamics, we would have a more compelling game while also mandating that our AI be able to handle more complex situations.

Another area the team felt they could improve if given a second chance was the planning stage in A-term. By putting off all implementation until the second term, the team was unable to discover issues until very late in the process. For example, the team envisioned a large map to have the player explore. However, it was not until after the map was loaded and the characters were appearing on the map that the team noticed that having a scrolling map would invalidate much of the existing code. Had the team built a game-window earlier, the problem may have been solved and the large map would have been preserved. Likewise, realizing that the game component was lacking would have come naturally, instead of coming as a surprise. As soon as

the team settled on some basic concepts and design choices in the first two weeks, coding should have begun.

Lastly, and the most evident of our mishaps was the ambition with which we initially pursued this project. As is the case with almost all project teams, they go in with an idea that is a little out of their reach. Over the course of the project they will do all they can to achieve that initial goal, but reality always sets in. This project was a clear case of that. We had great ideas for what this project could become, but when we thought of them, we really had no idea just how much work they would involve. It is always better to start with a reasonable workload and add on from there. That is what we should have done.

Overall, the project was a success. Despite minor and major setbacks, the team delivered a small, casual game, which demonstrated the social artificial intelligence middleware they created, for developers to use in future games.

Bibliography

BioWare. (n.d.). *Dragon Age: Origins*. Retrieved March 15, 2010, from http://dragonage.bioware.com/

Buckland, M. (2005). *Programming Game AI by Example.* Plano, Texas: Wordware Publishing Inc.

Microsoft. (n.d.). *XNA Creators Club Online - minjie*. Retrieved February 26, 2010, from XNA Creators Club Online - home: http://creators.xna.com/en-US/minigame/minjie

Microsoft. (n.d.). *XNA Creators Club Online - sprite sheet*. Retrieved February 26, 2010, from XNA Creators Club Online - home: http://creators.xna.com/en-US/sample/spritesheet

The Sims. (n.d.). Retrieved March 15, 2010, from http://thesims.ea.com/

Appendix A Script

Opening Sequence

Title Slide
Prince of Pride

Controls Slide
Prince of Pride Controls

Left Mouse Button Click:

- Selects objects in the game world
- Selects an action from the Action Bar
  (on leftmost side of User Interface)

Right Mouse Button Click:

- Moves Sefu to the targeted area

Holding down I Button displays current pride members status

Holding down U Button displays controls

Escape exits the game

Story Slide
The Story

You are the lion Sefu! You are a prince of the Madaha Pride! You have to assist the members of the pride in getting through a number of difficult situations. Once you have helped the lions through this period of hardship, you will be given a score based on how well you dealt with the problems and how well you can work in a team.

Sefu's Slide
Meet the Lions!

Sefu is the prince of the pride. He is the lion that you control.It is up to you to make Sefu who you want him to be.

Kojo’s Slide
Meet the Lions!

Kojo is the leader and the king of the pride. He is a harsh, mean ruler.All he ever does is go around roaring at the other lions. It makes them feel bad, which makes him feel good.

Amadi’s Slide
Meet the Lions!

Amadi is the warrior of the pride. He is the strong,yet gentle and kind lion that everyone wishes Kojo was. He helps prepare the lions for the hunt by exercising with them every day.

Binta’s Slide
Meet the Lions!

Binta is the energy of the pride. This little cub is full of life and has a positive attitude. She jumps for everyone's entertainment, but this can make her tired.

Transition Slide
Out in the African savannah after a successful night's hunt...

In-game Dialogue

Meat Giving
Before Meat is Given to Pride Members (Dialogue)
Kojo: Oh, it's a good life. Amadi and Zabia hunt and I eat.

Amadi: This is unfair. Kojo always has a full belly, while the rest of us beg for scraps.

Binta: Sefu help us get some meat before it's all gone.

Before Meat is Given to Pride Members (Description)

Kojo: Kojo looks ready to go eat some of the meat.

Amadi: Amadi is very frustrated.

Binta: Binta looks tired, but still hopeful.

After Meat is Given to Pride Members (Dialogue)

Kojo (after first piece of meat is taken): Hey! Save me some meat!

Kojo (after second piece of meat is taken): Stop! I want some meat!

Kojo (after last piece of meat is taken): Hey! I was going to eat that!

Amadi: Thanks Sefu! Now I won't be hungry!

Binta: Thanks Sefu! Now I won't be hungry!

After Meat is Given to Pride Members (Description)

Kojo (after first piece of meat is taken): Kojo is getting irritated.

Kojo (after second piece of meat is taken): Kojo is angered by your misbehavior.

Kojo (after last piece of meat is taken): Kojo is beyond mad.

Amadi: Amadi looks very pleased.

Binta: Binta looks very pleased.

Binta Missing

Before Cub is Found (Dialogue [Part1])

Kojo: It looks like the cub is missing ... too bad. If you're expecting my help you're crazy.

Amadi: Please help me look for the child. She couldn't have gone far.

Before Cub is Found (Dialogue [Part2])*

Kojo: Leave me alone! If the cub ran away it's her fault!

Amadi: Don't give up. Keep looking.

Before Cub is Found (Dialogue [Part3])*

Amadi: I thought I saw something stirring in the bushes, but my eyesight is fading. Look for me.

Before Cub is Found (Description)

Kojo: Kojo looks mean and mad like he always does.

Amadi: Amadi looks really worried.

After Cub is Found (Dialogue)

Kojo: It appears you found the cub. What a hero ...

Amadi: Thank you so much for your help.

Binta: Sefu, you saved me!

After Cub is Found (Description)

Kojo: Kojo looks mean and mad like he always does.

Amadi: Amadi looks happy and relieved.

Binta: Binta is very grateful.

*this dialogue may not appear if Binta is found before this set of hints is triggered

Medicine Giving

Before Medicine is Given to Binta (Dialogue)

Kojo: What is Binta complaining about?

Amadi: I think Binta may have eaten some bad meat. She looks like she could use some medicine. Here give her this.

Binta: I feel very weak. I need medicine.

Before Medicine is Given to Binta (Description)

Kojo: Kojo looks indifferent.

Amadi: Amadi needs your help.

Binta: It looks like Binta will not have much time if she doesn't get medicine soon.

After Medicine is Given to Binta (Dialogue)

Kojo: What do you want?

Amadi: You are a great lion Sefu.

Binta: This medicine will be very helpful. Thank you Sefu.

After Medicine is Given to Zabia (Description)

Kojo: Kojo looks indifferent.

Amadi: Amadi looks admirably at you.

Binta: Binta is already getting better.

Skull Giving

Before Skull is Given to Kojo (Dialogue)

Kojo: I'm not being mean because I feel like it. I would be nice, but no one has ever has ever appreciated how hard I work to keep us safe.

Amadi: Give Kojo the skull. Let him know we care for him.

Binta: I found an antelope skull, it looked pretty cool, but I'm not going to tell you where I hid it.

Before Skull is Given to Kojo (Description)

Kojo: Kojo looks disappointed, but ready to change.

Amadi: Amadi sees an oppurtunity to change the pride for the better.

Binta: Binta is very excited.

After Skull is Given to Kojo (Dialogue)

Kojo: Thank you so much, Sefu. You have no idea how much I appreciate this.

Amadi: You are a great friend Sefu.

Binta: We're one big happy family!

After Skull is Given to Kojo (Description)

Kojo: Kojo finally feels appreciated.

Amadi: Amadi has no more worry.

Binta: Binta looks ready to celebrate.

Moon Transitions

Quarter Moon

If everyone is given meat
Everyone will be going to bed with a full belly.
If everyone is not given meat
It appears some of the pride will be going to bed hungry. But Kojo won't.

Half Moon

If Binta is found
Binta is safe. Everyone can breath easier.
If Binta is not found
Binta is still missing. No one will sleep tonight.

Three Quarter Moon

If Binta is given medicine
Binta is feeling much better. She is back to her old self.
If Binta is not given medicine
Binta is still sick. She might not make it through the night.

Full Moon

If skull is given to Kojo
The pride is changing for the better.\n Kojo is now the king everyone had thought he would be.

If skull in not given to Kojo
"The pride is still sad, and Kojo is still mean."

General Descriptions

Tree
The only tree for miles.
Rock
The rock that this pride's land is based around.
Bush
Some of the brush on the savannah.
Meat
The meat from last night's hunt.

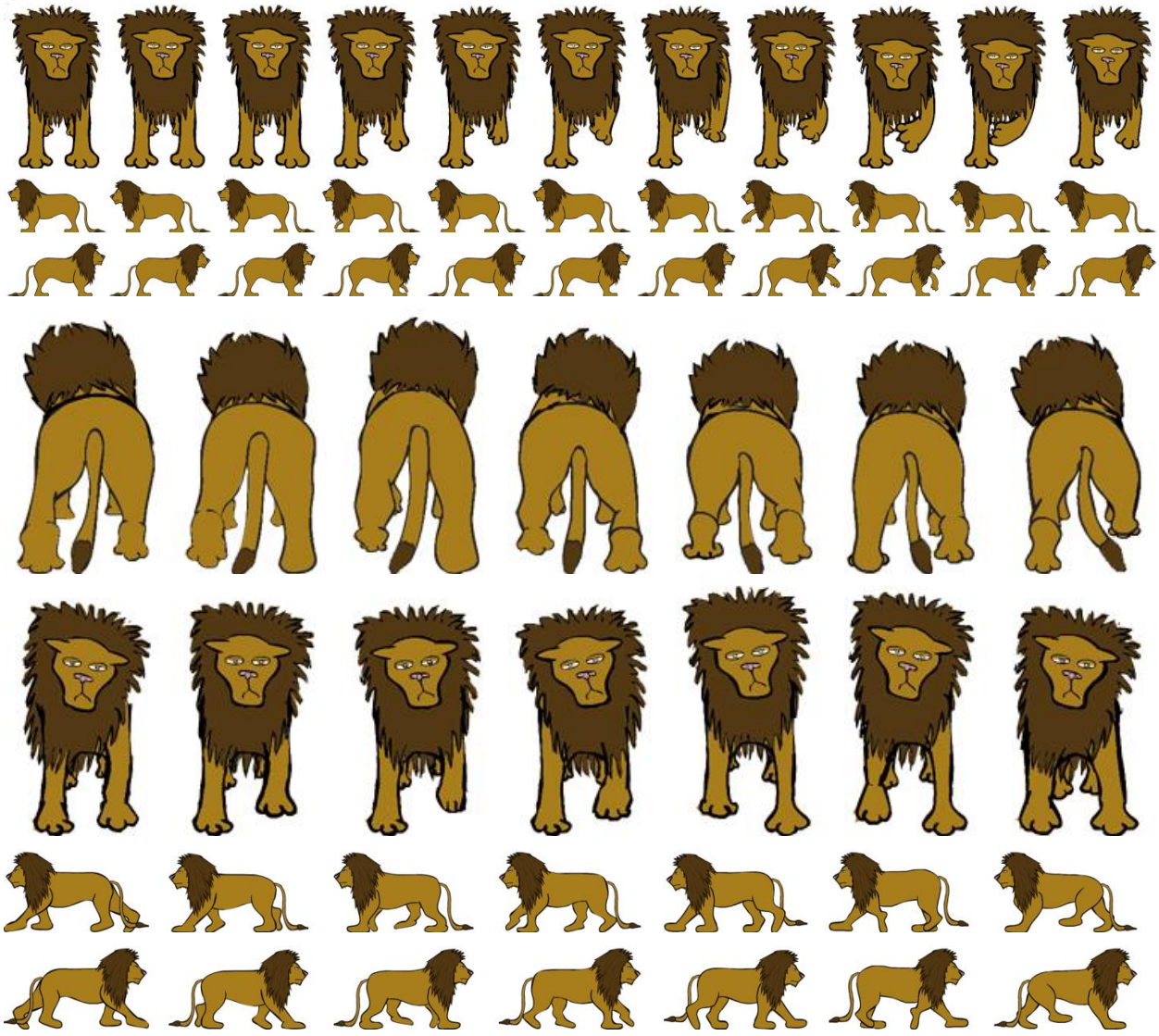
Appendix B Final Design Review

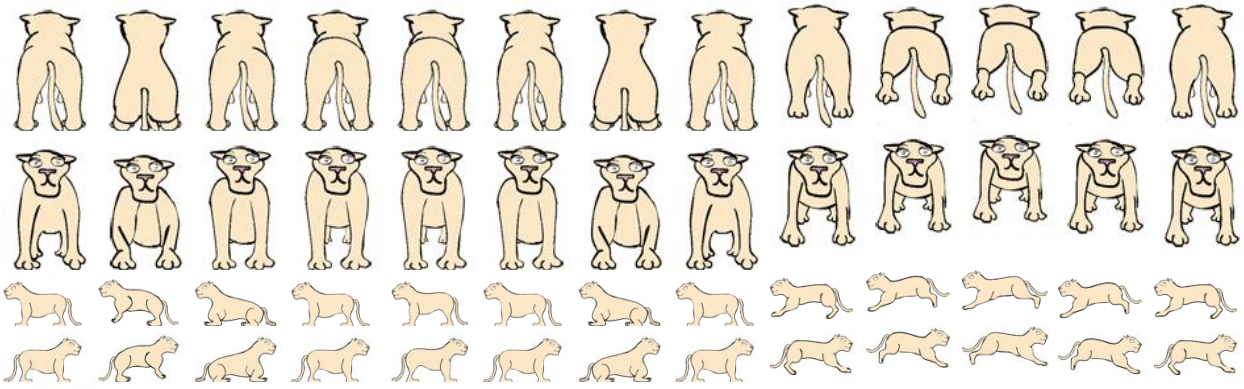
To view our Final Design Review, please refer to the attached document, “SocialAI_FDR”. This paper outlines the plan for the implementation and testing of the project that was agreed upon at the end of the first term of this project.

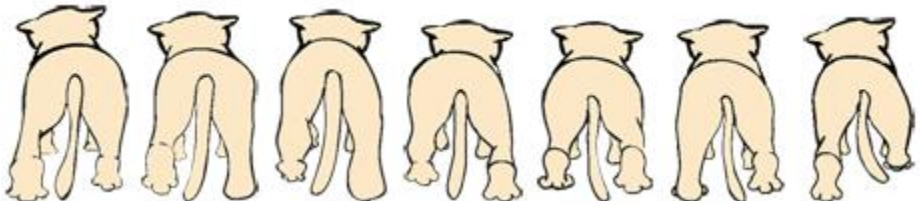
Appendix C Code

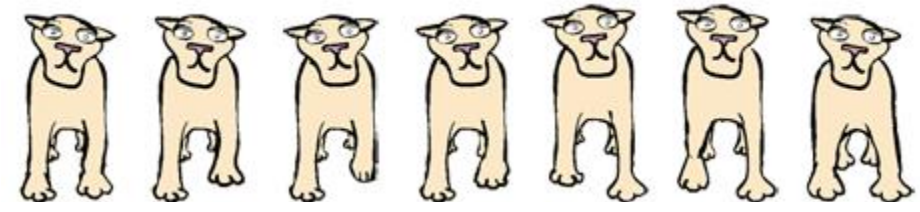
To view the source code for this project, please refer to the document named "Prince_of_Pride_Code". It is recommended that the code be viewed in WordPad with wordwrap off. This is the simplest textual viewing device that allows for the most readability. It is also recommended, since the document is long, that Ctrl-F (Find) be used when looking for a specific instance in the code.

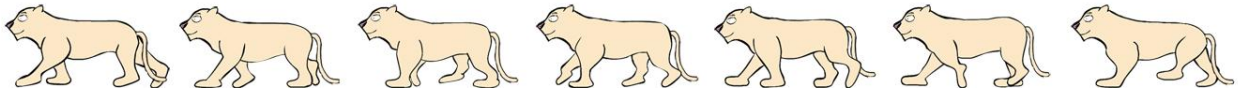
Appendix D Art

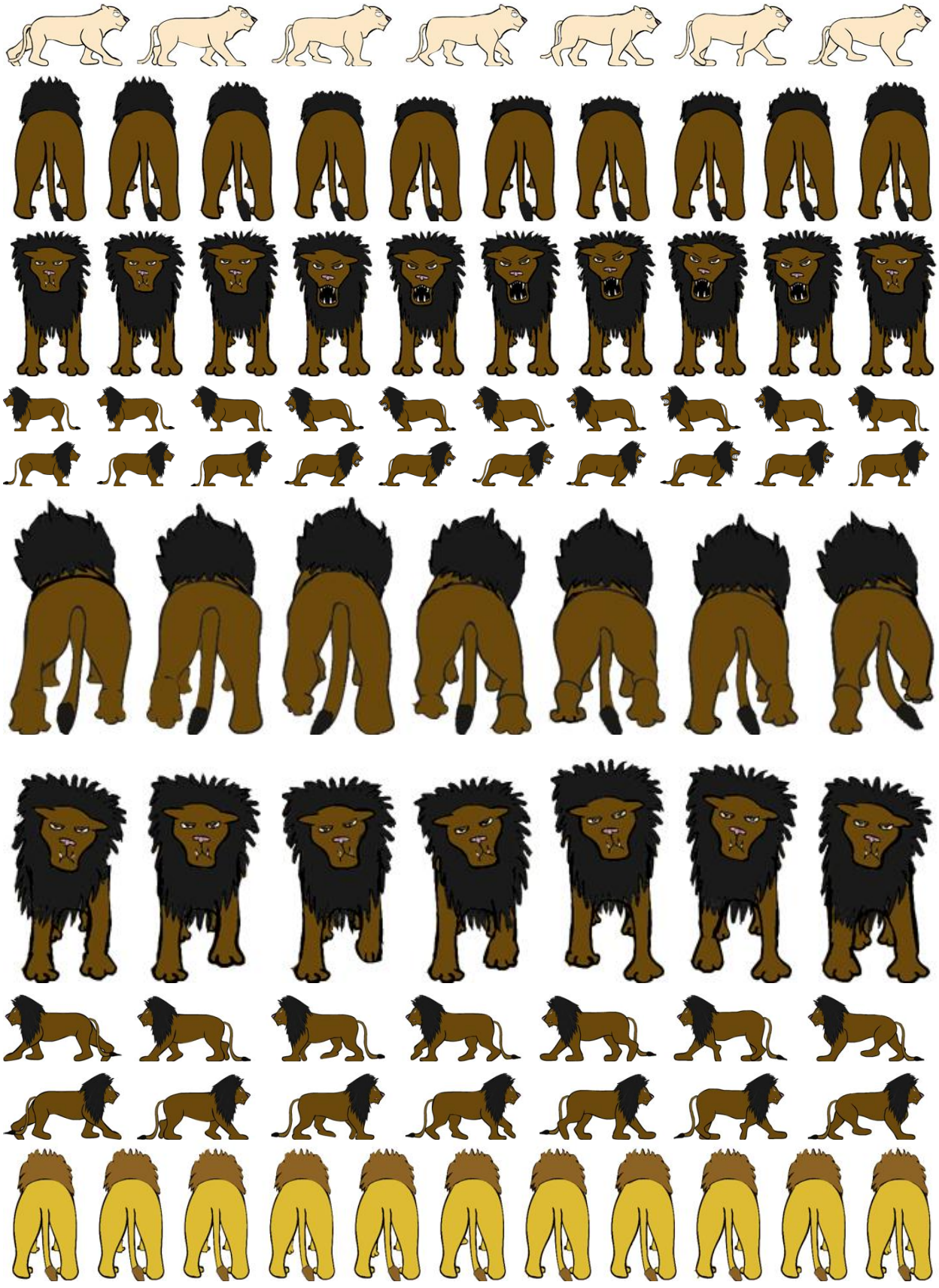


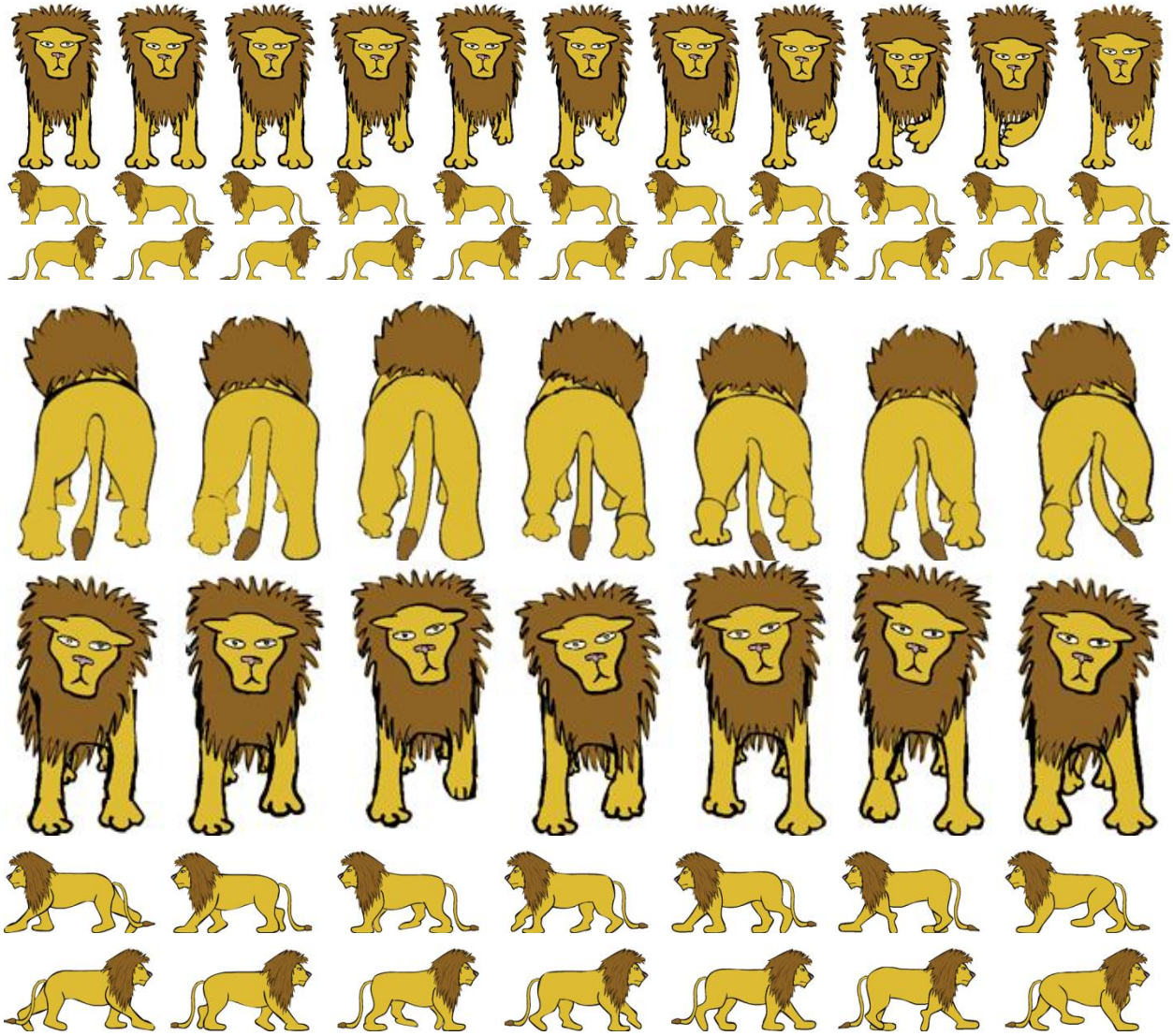


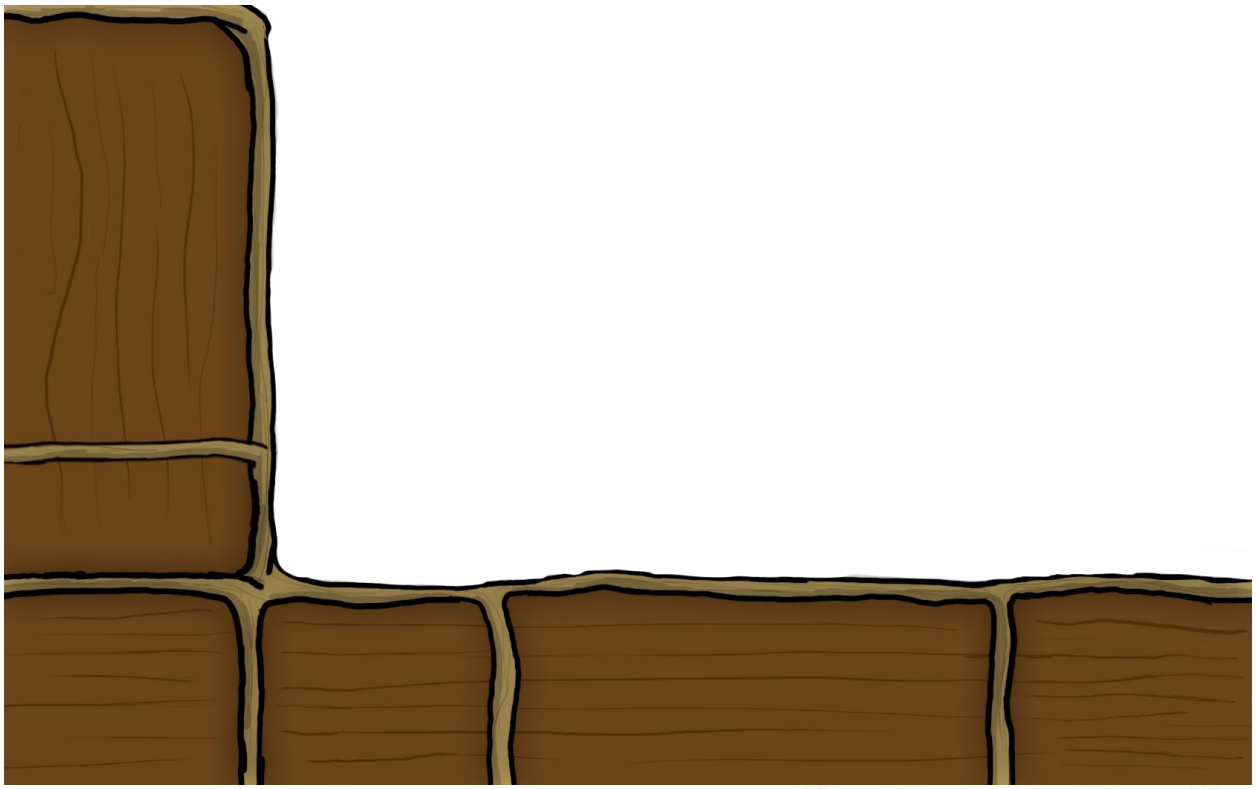


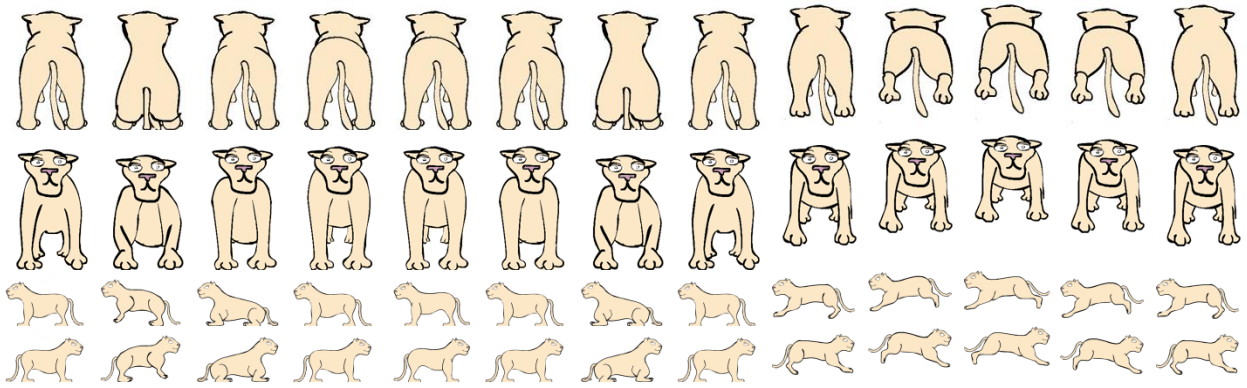


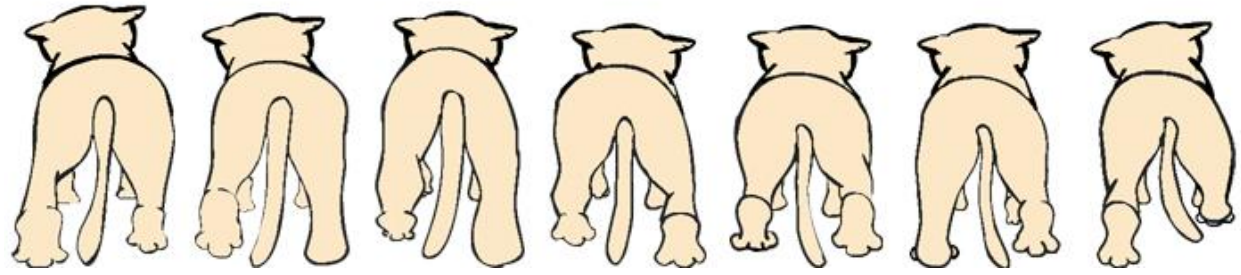


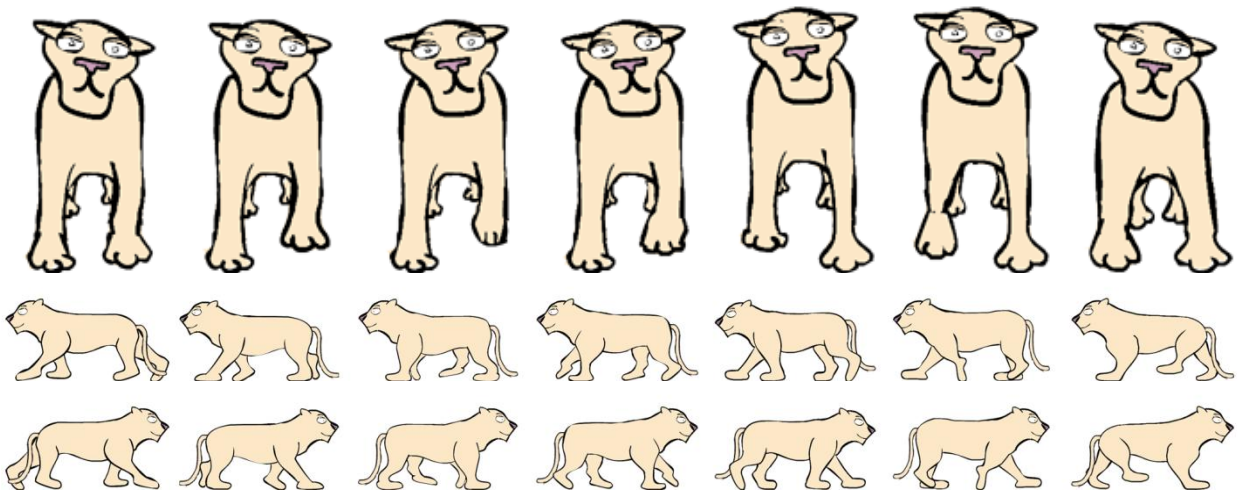












Appendix E Testing

Problems Identified from Testing

- Meat did not disappear when taken
- When selecting an action then changing target and choosing an action, the lion performs the first action.
- The time for the cub missing event was too long.
- Not sure what they had selected
- Binta needs to get sadder when she gets sick.
- Giving the skull did not work.
- Kojo should get upset when not given the skull.
- Responses did not change when the skull was given to someone other than Kojo.
- Items were not being removed from the map.
- Skipping moon sequences caused problems with events starting after going back to the game.

Comments from Testers

- All testers were casual gamers
- Make Sefu faster
- Make it so that you do the action that you chose last.
- Make it so the characters are unhappier.
- Make the text clearer.
- Make it clearer what you need to do.
- Make it so Binta appeared from behind the bush she was supposed to be behind.
- Add another character (Zabia).
- Older testers, ages 16 and up, stated that the game was too simple
- Younger testers, ages 16 and under, stated that the game was fun
- Game was easy to learn
- When asked what they thought the main purpose for the game was, teamwork was chosen 60% of the time and investigatory was chosen 30% of the time
- Older testers said the game was too short
- Younger testers said the game was a good length
- When asked whether they saw this game as just another game or an educational tool, all older users said an educational tool while 80% of younger testers said game

Appendix F User Interface Asset List

UI Asset List

Done?	Filename	Amount
x	actionbar	1
x	messagebar	1
	messagebox	
x	itembar	1
x	indivinfo	1
x	prideinfo	1
x	picwindow	1
	Total Assets:	6

Appendix G Audio Asset List

filename
lionsnarl-stereo.wav
liongrowl.wav
lion-shortroar2.wav
lionsnarl.aif
lion-roar-scare.wav
lion-irritated.wav
lion-shortroar.aif
lion-shortroar-fierce.wav
lion-cough.wav
lion-snore.wav
lioncub-whine.wav
lionroar-reverb.wav
lion-battlecry.wav
lion-roar-multiple.wav
lion-roar-upset.wav
lion-hiss.wav
lion-cub-teen-roar.wav
lion-cub-baby-whine.wav

Appendix H Environment Asset List

Environment Asset List

Done?	File Name	Amount
	Rocks	
x	rock sprite small	3
x	rock sprite medium	2
	Trees	
x	tree sprite large	1
	Grass	
x	grass bunch sprite	4
	Bushes	
x	bush sprite	4
	Backgrounds	
x	grassland	1
	Total Necessary Assets:	15

Appendix I Character Asset List

done?	filename	colors
	GENERAL ACTIONS:	
x	lion-walk-01-f	10
x	lion-walk-02-f	10
x	lion-walk-03-f	10
x	lion-walk-04-f	10
x	lion-walk-05-f	10
x	lion-walk-06-f	10
x	lion-walk-07-f	10
x	lion-walk-01-b	10
x	lion-walk-02-b	10
x	lion-walk-03-b	10
x	lion-walk-04-b	10
x	lion-walk-05-b	10
x	lion-walk-06-b	10
x	lion-walk-07-b	10
x	lion-walk-01-l	10
x	lion-walk-02-l	10
x	lion-walk-03-l	10
x	lion-walk-04-l	10
x	lion-walk-05-l	10
x	lion-walk-06-l	10
x	lion-walk-07-l	10
x	lion-walk-01-r (reversible)	10
x	lion-walk-02-r (reversible)	10
x	lion-walk-03-r (reversible)	10
x	lion-walk-04-r (reversible)	10
x	lion-walk-05-r (reversible)	10
x	lion-walk-06-r (reversible)	10
x	lion-walk-07-r (reversible)	10
x	adultmale-stand-f	4
x	adultmale-stand-b	4
x	adultmale-stand-l	4
x	adultmale-stand-r (reversible)	4
x	adultfemale-stand-f	3

x	adultfemale-stand-b	3
x	adultfemale-stand-r	3
x	adultfemale-stand-l	3
x	cub-stand-f	3
x	cub-stand-b	3
x	cub-stand-l	3
x	cub-stand-r	3

SPECIFIC ACTIONS:

x	adultmale-roar-01-f	4
x	adultmale-roar-02-f	4
x	adultmale-roar-03-f	4
x	adultmale-roar-04-f	4
x	adultmale-roar-05-f	4
x	adultmale-roar-01-b	4
x	adultmale-roar-02-b	4
x	adultmale-roar-03-b	4
x	adultmale-roar-04-b	4
x	adultmale-roar-05-b	4
x	adultmale-roar-01-l	4
x	adultmale-roar-02-l	4
x	adultmale-roar-03-l	4
x	adultmale-roar-04-l	4
x	adultmale-roar-05-l	4
x	adultmale-roar-01-r (reversible)	4
x	adultmale-roar-02-r (reversible)	4
x	adultmale-roar-03-r (reversible)	4
x	adultmale-roar-04-r (reversible)	4
x	adultmale-roar-05-r (reversible)	4
x	adultmale-swipe-01-f	4
x	adultmale-swipe-02-f	4
x	adultmale-swipe-03-f	4
x	adultmale-swipe-04-f	4
x	adultmale-swipe-05-f	4

x	adultmale-swipe-01-b	4
x	adultmale-swipe-02-b	4
x	adultmale-swipe-03-b	4
x	adultmale-swipe-04-b	4
x	adultmale-swipe-05-b	4
x	adultmale-swipe-01-l	4
x	adultmale-swipe-02-l	4
x	adultmale-swipe-03-l	4
x	adultmale-swipe-04-l	4
x	adultmale-swipe-05-l	4
x	adultmale-swipe-01-r (reversible)	4
x	adultmale-swipe-02-r (reversible)	4
x	adultmale-swipe-03-r (reversible)	4
x	adultmale-swipe-04-r (reversible)	4
x	adultmale-swipe-05-r (reversible)	4
x	adultmale-jump-01-f	4
x	adultmale-jump-02-f	4
x	adultmale-jump-03-f	4
x	adultmale-jump-04-f	4
x	adultmale-jump-05-f	4
x	adultmale-jump-01-b	4
x	adultmale-jump-02-b	4
x	adultmale-jump-03-b	4
x	adultmale-jump-04-b	4
x	adultmale-jump-05-b	4
x	adultmale-jump-01-l	4
x	adultmale-jump-02-l	4
x	adultmale-jump-03-l	4
x	adultmale-jump-04-l	4
x	adultmale-jump-05-l	4
x	adultmale-jump-01-r (reversible)	4
x	adultmale-jump-02-r (reversible)	4
x	adultmale-jump-03-r (reversible)	4
x	adultmale-jump-04-r (reversible)	4
x	adultmale-jump-05-r (reversible)	4

x	adultfemale-rub-01-f	3
x	adultfemale-rub-02-f	3
x	adultfemale-rub-03-f	3
x	adultfemale-rub-04-f	3
x	adultfemale-rub-05-f	3
x	adultfemale-rub-01-b	3
x	adultfemale-rub-02-b	3
x	adultfemale-rub-03-b	3
x	adultfemale-rub-04-b	3
x	adultfemale-rub-05-b	3
x	adultfemale-rub-01-l	3
x	adultfemale-rub-02-l	3
x	adultfemale-rub-03-l	3
x	adultfemale-rub-04-l	3
x	adultfemale-rub-05-l	3
x	adultfemale-rub-01-r (reversible)	3
x	adultfemale-rub-02-r (reversible)	3
x	adultfemale-rub-03-r (reversible)	3
x	adultfemale-rub-04-r (reversible)	3
x	adultfemale-rub-05-r (reversible)	3
x	adultfemale-feed-01-f	3
x	adultfemale-feed-02-f	3
x	adultfemale-feed-03-f	3
x	adultfemale-feed-04-f	3
x	adultfemale-feed-05-f	3
x	adultfemale-feed-01-b	3
x	adultfemale-feed-02-b	3
x	adultfemale-feed-03-b	3
x	adultfemale-feed-04-b	3
x	adultfemale-feed-05-b	3
x	adultfemale-feed-01-l	3
x	adultfemale-feed-02-l	3
x	adultfemale-feed-03-l	3
x	adultfemale-feed-04-l	3
x	adultfemale-feed-05-l	3

x	adultfemale-feed-01-r (reversible)	3
x	adultfemale-feed-02-r (reversible)	3
x	adultfemale-feed-03-r (reversible)	3
x	adultfemale-feed-04-r (reversible)	3
x	adultfemale-feed-05-r (reversible)	3
x	cub-roll-01-f	3
x	cub-roll-02-f	3
x	cub-roll-03-f	3
x	cub-roll-04-f	3
x	cub-roll-05-f	3
x	cub-roll-01-b	3
x	cub-roll-02-b	3
x	cub-roll-03-b	3
x	cub-roll-04-b	3
x	cub-roll-05-b	3
x	cub-roll-01-l	3
x	cub-roll-02-l	3
x	cub-roll-03-l	3
x	cub-roll-04-l	3
x	cub-roll-05-l	3
x	cub-roll-01-r (reversible)	3
x	cub-roll-02-r (reversible)	3
x	cub-roll-03-r (reversible)	3
x	cub-roll-04-r (reversible)	3
x	cub-roll-05-r (reversible)	3
x	cub-roar-01-f	3
x	cub-roar-02-f	3
x	cub-roar-03-f	3
x	cub-roar-04-f	3
x	cub-roar-05-f	3
x	cub-roar-01-b	3
x	cub-roar-02-b	3
x	cub-roar-03-b	3
x	cub-roar-04-b	3
x	cub-roar-05-b	3

x	cub-roar-01-l	3
x	cub-roar-02-l	3
x	cub-roar-03-l	3
x	cub-roar-04-l	3
x	cub-roar-05-l	3
x	cub-roar-01-r (reversible)	3
x	cub-roar-02-r (reversible)	3
x	cub-roar-03-r (reversible)	3
x	cub-roar-04-r (reversible)	3
x	cub-roar-05-r (reversible)	3
	LION FACES:	
x	amadi-sad	1
x	amadi-happy	1
x	zabia-happy	1
x	zabia-sad	1
x	binta-happy	1
x	binta-sad	1
x	kojo-angry	
x	kojo-happy	
	TOTAL ASSETS REQUIRED:	806