PROJECT DAO

Iridium – 20 million leagues above the sea

Design Document

Richard Fredriksson, Robert Krantz, Olof Millberg, Mikael Nilsson, Fredrik Wendt

3/5/2009

A game built with Microsoft XNA focusing on player-AI interaction.

Document Change History

Document changes in version 0.3

Date: 2009-03-04

Section about Gameplay

Section about the AI

Added subsections to Code

Evaluation – Details and results from initial test

Document changes in version 0.2a

Date: 2009-02-12

Initial Release

Table of Content

Introduction 5
- Background 5
- Design Goals 5
  - Requirements for the game: 5
  - Design 6

Gameplay 9

AI 10
- Agents 10
  - Memory 10
  - Mood 10
  - Emotions 10
  - Needs 11
  - Personality traits 12
  - Skills 12
- Actions 12
- AI update 13
  - Sense Graph 13
  - AI-Event system 13
  - Find Actions 13
  - Execute 14

Architecture 15
- Overview 15
- Data 15
  - Physical Memory 15
  - Primary Memory 15
- Code 16
  - Core 16
  - Scene Graph 16
  - Input 16
  - Resource Manager 16
  - Interface 16
  - Message Passing / Event System 16
  - Console .... Error! Bookmark not defined.
  - Camera 17
  - AI 17

Operation 18
- User types 18
- Installation 18
- Licensing 18

Development Plan 18
- Design document version .... Error! Bookmark not defined.

Project Report .... Error! Bookmark not defined.
Swedish Game Awards .... Error! Bookmark not defined.
System versions .... Error! Bookmark not defined.
Project Plan .... Error! Bookmark not defined.
Period three .... Error! Bookmark not defined.
Period four .... Error! Bookmark not defined.
Evaluation 19
Appendices 20
Low Fidelity Test Feedback 20
Open Issues 21
Glossary 21
Bibliography 21
Reference 21

Introduction

Background

Iridium is a real-time strategy game set in space. It is developed in XNA, for the Windows platform as well as, potentially, XBOX 360. In the game the player assumes the role of captain of a spaceship, commanding a small crew and completing missions in order to gain money and experience.

The game is played in single player on a local machine. The game play is aimed at casual gamers and focus is placed on creating interesting behavior the player can observe and act upon rather than fast paced action.

The main feature that separates Iridium from other games in the genre is the control system used to control the ship. The player does not have direct control of the crew - specifically telling a character to stand on a specific spot or talk to another character will not be possible. Instead the player must control the situation by assigning positions and work shifts to the different crew members or, in worst case, give orders to contain or break up volatile situations. Observing the crew and how the members interact with each other as well as how they react to situations imposed on them, will help the player in running the ship and crew smoothly.

Design Goals

This section details the functional and non-functional requirements for the game as well as the design goals.

Requirements for the game:

Here is a detailed list of the requirements for the project.

Functional requirements:

- The game must feature a graphical 3D view.
- The view camera must have pan, zoom and rotate capabilities.
- The graphics in the game should be scalable.
- The game should have support for utilizing Lua scripting language.

- The game must contain models and textures for all physical entities.
- The game must contain animations.
- The game should support blended animations.

- The game must feature an interface for character creation.
- The game must contain an interface system for recruiting and replacing crew members.
- The game must feature a control system for controlling crew through assigning tasks. There should be no direct control over each individual character available.
- Large parts of the game should be controlled through a context-sensitive menu with different options available depending on where the menu is activated.

- The game must contain AI for in-game characters.
- The game AI must be able to simulate a subset of human emotion and interaction.

- The game AI must feature path finding functionality for the crew members.

- The game should contain an introductory story.
- The game must contain at least 2 missions for the player to choose and play through.
- During missions (ship view) the game must play in real time.
- The game must contain at least 1 critical situation forcing the player to act upon unforeseen circumstances.
- The player character should be able to gain experience and improve abilities.
- The game must contain functionality for the player to improve and expand his/her ship.
- The difficulty of the game should be scalable.

- The game must contain music.
- The game music should be available at different qualities.
- The player should be able to activate/deactivate the music as well as raise and lower the volume of the music.
- The game must contain sound effects.
- The game should have 3d sound.

Non-Functional:

- The game must run at an even level of at least 30 frames per second. (on medium hardware, to be specifically specified later)
- The interface must be easy to understand and use to its full potential.
- The code should be easily extendable.

Design

This section details the design goals we have set within the different areas of the project.

Programming

Several goals have been set for the code of the project; AI, game logic, sound e.g. are all aspects that have specific requirements. The AI will control the characters in the game; it will supply each character with a behavior that is based on its personality and the situation. The game logic is supposed to control the different game states and also takes care of the crew simulation. The sound component will support will control all the special sound effect in the game. It will utilize 3D sound in order to select what sounds that will be played. Background music will be streamed in order to minimize resource utilization. The framework of the game will use the LUA scripting language to enable live editing of the game play parameters. It will also implement a message system in order to handle internal communication. The game world will be implemented through a scene graph and in order to make it possible for the characters in the game to find their way around the game world, a path finding algorithm based on A* is utilized.

Graphics

The lighting in the game will consist of diffuse light as well as spotlight. As for the shadows; shadow volumes is used in order to create nice looking shadows. Also shaders will be used in order to make the game look more like the visual style we have chosen.

Assets
In order to make the visual style possible, many assets are needed. There will be a user interface and this will require textures. Models of the space ship as well as all its furniture and interior details are needed. Also characters in all their variants must be created; these and all the other models also require textures. The sound assets will consist of music and effects. As for the characters they will also need statistics, name and possibly experience. Also enemies and monsters must be created though due to the limited time frame and the amount of time they appear in the game, a lesser detail will be required for these.

Input
The controls in the game will be mouse and keyboard, there will be keyboard shortcuts. The UI of the game should be easily converted to X-Box and they menus will be context sensitive.

Gameplay
In the game there will be a menu for recruiting new crew members to the ship as well as fire existing crew members, the player will also be able to view the CV of all the available and hired crew members. The player will be able to select which crew members that are stationed at a certain place. He will also have a crew rooster at his disposal to aid him in doing the appropriate choices. The mission selection menu will consist of a mission length selector as well as a mission difficulty selector, important to notice is that the user will select the amount of time he or she will spend playing that mission. Good for controlling the amount of time spent playing the game. The player will also be able to choose between two different missions. During a mission there might occur minor and/or major critically situations. These help keep the players challenged from time to time.

Programming

- AI
  - Behavior
  - Personality
- Game logic
  - Game states
  - Crew simulation
- Sound
  - 3D-sound
  - Streaming
- Framework
  - Lua support
  - Message system
- Scene graph
- Path finding
- Lua

Graphics

- Lighting
  - Diffuse
  - Spotlight

- Shadows
    - Shadow volumes
- Shading
    - Style shader

Assets

- Textures
    - Models
    - User interface
- Sounds
    - Effects
    - Music
- Models
    - Ship
    - Interior
    - Interior detail
    - Character models
    - Furniture
- Text
- Characters
    - Ship crew
        - Name
        - Stats
        - Experience
- Enemies
    - Monsters
    - Pirates

Input

- UI
- Controls
    - Context sensitive
    - Keyboard shortcuts

Gameplay

- Recruiting
    - CV
    - Roster
- Mission selection
    - Length
    - Difficulty
- Missions
    - Minor situations
    - Critical situations

Gameplay

- As Captain of a starship you seek your destiny in the galaxy while commanding and managing your crew.

The core game play consists of you as the player plays the captain, which tries to control a spaceship. The captain has no direct control of the spaceship so he or she must rely on the crewmembers of the ship to carry out the orders given. At the beginning of the game the captain must recruit the crew for the ship via the crew selection menu. He also selects the position of the crew members of the ship. Most of the time spent playing will be when the ship is travelling through space. The player which takes the role of the captain must keep the order in the ship despite events happening during missions. Event could e.g. be that crew members start fighting each other, aliens attack or that your ship is infected with a virus. These events require the captain to act and it is up to you as the player to choose how to solve the problems. There will be several different solutions to a common problem and this makes the gameplay more varied.

When the captain is not handling events he has other duties to attend to, one such would e.g. be to set the schedule of the ship.

The scheduling system is used by the player to specify the everyday tasks and responsibilities of the crew. In this system the player can set up who among the crew will work during the different shifts and what position or priority each crew member has in his field of expertise. The system consists of one window for each type of worker in the ship. Two examples are gunners and technicians. Each of these two groups has their own shift scheduler and priorities.

Whenever an event occurs, such as the ship being attacked, the player can change the ship's alert-level, which will make all the crew-members act according to the tasks assigned them in the scheduler. Most situations will, however require direct actions taken by the player through the context menu.

AI

The crew needs to work mostly independently and answer to orders. But even an order could be ignored. The most important aspect is the interaction between the crew members. There for the crew also needs a memory of how they feel towards everyone else.

Agents

Each agent has a number of variables that is the basis for their personality and functionality.

Memory

In order to have a more rationale behavior the agents are expected to act differently when interacting with someone they like or dislike. Also, when confronted for the second time with a situation it would be natural for the agent to receive less impact of this the second time. In order to facilitate this behavior each agent will track four variables of every other agent, situation and object that they have encountered.

Each agent will remember:

- Happy-for or resentment towards the target
- Gloating or pity towards the target
- Admiration or reproach towards the target
- Gratitude or anger towards the target

Mood

Each agent will have a variable mood which is the overall feelings of the agent at a point. Any change to his emotions will either have a positive or negative effect on the agent's mood.

The reason for tracking the mood is that agents will only over the course of hours have significant impacts on the mood unlike for emotions. The mood is a long term personality modifier.

The mood is used to determine whether or not an agent can do a certain action. A very good mood will have the agent act differently (pick different actions) compared to an agent with a bad mood. It will also play a part in calculating the current need of an agent.

Emotions

As mood is the slow moving part of the agent's current state emotions is the quick moving. Emotions span across several variables and can jump from minimum to maximum in the matter of seconds.

Each agent can experience these emotions:

- Pleased or displeased
- Joy or distress
- Hope or fear
- Satisfaction or fears-confirmed
- Relief or disappointment
- Approving or disapproving
- Pride or shame
- Gratification or remorse

Emotions will together with the mood determine what actions the agent can opt to execute. An agent in a very good mood, but currently very displeased, would act differently from one in a good mood and very pleased.

The basis for both emotions and the underlying emotions for the memory is all derived from work done by Ortony, Clore, and Collins with the OCC model [OCC]. The model is one of emotion synthesis.

Needs

Every agent has needs that are determined depending on the agent's current mood and emotions. The needs are based on Maslow's hierarchy of needs [MASLOW]. The different traits will modulate how a certain emotional impact will affect an agent.

Physiological

The most basic set of needs an agent exhibits. Most of them are obvious as they are literally requirements for an agent to survive. Also most of them are not needed in a game scenario. The ones used are sleeping and eating.

Each agent will over the course of a day slowly grow the physiological need. These needs will have the least focus in the game as it really is the least interesting one for the game concept. Agents will when they wake up take care of eating and do so at set times each day. In the evening they will go to bed. The only real obstruction in this plan is if the player would force his agents to wake up during the night, while the agent would fall back into his normal patterns it would have an impact on his mood. This would always be a tradeoff between having agents in a better mood compared to taking care of a situation in the middle of the night.

Safety

Only after having their physiological needs satisfied (inside of a threshold) the agents will see too their safety needs. These needs include personal and financial security and health and wellbeing. In the game the most important of these needs that will play a part in the concept is the wellbeing of each agent and how secure they feel.

Affection

When both safety and physiological needs are satisfied, the agents will start to worry about love, affection and belonging. This could entail in the game simply being social, talking and hanging out with the agents the agent likes.

Esteem

When all the prior needs are taken care of, the agent will need self-esteem, respect from others and raising their self-confidence. This could be work related as respect on the ship often ties into the work being done, and the importance of the mission.

Self-Actualization

With every other need satisfied, the agent can concentrate on his need to be and do that which he was born to do. For a musician that would be to play music, an artist would paint, and a slacker would slack.

Personality traits
The personality for each agent determines how he is going to act and react to other agents, situations and objects. Each trait is invariable and is based on the work done by Costa and McRae and the five factor model of personality, known as OCEAN (each letter corresponding to a trait) [OCEAN].

Neuroticism
A high grade of neuroticism personifies an agent that often worries, is nervous and insecure. On the opposite end (a low score) is an agent that is calm, relaxed and secure.

Extraversion
Basically how social and outwards an agent is. High grade corresponds to social, talkative and optimistic. Low grade is an agent that is reserved, quiet and unoptimistic.

Openness
This trait has to do with the agents will to explore and try new things. A high grade gives an agent that is very curious, creative and has a broad set of interests. A low grade corresponds to being conventional, very down to earth and having a small set of interests.

Agreeableness
The agents will by this factor be more or less willing to help each other and follow orders. A high grade of this trait means that the agent is good-natured, trusting and very helpful. On the opposite side a low grade is an agent that is cynical, rude and very uncooperative.

Conscientiousness
By large determines how organized and precise the agent is. High grade is reliable, hard-working and punctual. Low grade is aimless, lazy and negligent.

IQ (stand alone)
A separate personality trait is an IQ which modulates how fast (or slow) the agent will learn skills.

Skills
EQ - social skill, rank on ship and Ship type skills.

Actions
Actions can be any number of things that the agents should be able to do. As an example an insult could be an action. This action would be available after the agent has established that there is someone in the vicinity that he dislikes, and that the character has the appropriate emotional state of mind. E.g. he might need to be pretty sad and somewhat insecure before he starts insulting people.

The game will focus on character interaction and a lot less about things like eating and sleeping. Most important will be actions that directly are done between two agents like them talking to each other.

Possible actions are taking care of all the jobs that need to be done (navigator, gunner, engineer), talking to other characters (chat, insult, charm, flirt), and lots more.

A lot of actions apply to several situations but in different ways. If a person is talking to someone they like, this is very different from someone doing the same action but talking to someone they don't like.

AI update

The AI is very central and important for the game, there for a lot of CPU cycles are dedicated to this task. For each agent the game will find a suitable action, and execute this action and calculate what impact his and other agents' actions have on the character.

Sense Graph

The sense graph is used for all the agents' perception. Storing all the information of what is happening in the entire ship in a format that each agent can perceive.

The sense graph will register all the actions and let them propagate through the ship using a smell that will decay for each node it travels. This way, actions will have a proximity that will be affected, and all the agents can react to what is happening around them.

AI-Event system

In order to broadcast messages and receive messages an event system will be in place. Some messages will be global, some only for a room, and some directly sent to another character. The events are used in the appraisal method.

Find Actions

If an agent is idle or doing something that will allow for a second concurrent action, he will look for new actions. To find a suitable action a goal oriented behavior is used. It will for a list of actions, and an agent, find the best action for the character by calculating how well of the agent is after every possible action.

Goal oriented behavior

The first step of finding an action is to perceive the room. This way a number of different things can be discovered, is there a person in the room to pick a fight with, is there anything in the room to make me happy etc.

Possible actions

To find the possible actions for an agent a tree is stored with all the actions. The actions are leafs and the nodes are different conditionals. In order to reach an action the current character might need to be a certain amount of happy or sad. Also from the sense graph a number of conditionals are received which are used to know what parts of the tree to traverse.

With all the possible actions found, with an intended target (other character, machines, point of interest etc.) the goal oriented behavior algorithm runs.

The "best" action

In order to find the best action, for each action the algorithm calculates the change it will have on the agent, and the amount of well being that it results in. Also for each action the score will be modulated by the amount of time involved in carrying out the action, this can be predefined or calculated from the time it takes to move somewhere.

In order to make the game more playable or interesting, the best action won't always be from the perspective of the agent. It could be to force an agent to follow an order from the player, or to have more stupid actions be chosen in order to make it more fun.

When a new action is found and chosen this is stored with the agent and the process proceeds.

Execute

The second part of the AI loop is the execute step.

Run action

Each action the agents have is started (and if already started the process proceeds). The run action will do the animation, logic, path finding and steering behavior.

Appraisal

In the next step the process will do the appraisal. The aim of the appraisal is to go through all the reactions of the agent. Each time something around the agent happens to others or to him, he can react to it. The reaction will modify his current mood, his emotions and his memory towards others. This process is also modulated by the character personality. With the mood and emotions recalculated each need will also be updated for the agent.

The appraisal and emotions are both a part of the OCC model [OCC] by Ortony, Clore, and Collins.

Architecture

Overview

Iridium is a single-player game and the system is therefore stand-alone with no network support. It is primarily developed for the PC platform under the Windows operating system. Since the XNA framework is used there is a possibility of porting the game to the Xbox 360 platform. The game will require a computer with a graphics card supporting Pixel Shader 3.0.

Most of the game logic will be written in the scripting language Lua. This means that a kind of API will be written in C# for access to XNA as well as various data structures and algorithms.

The player will control the game mainly with the mouse (and possibly gamepad). Direct inputs will include moving the mouse to the edge of the screen or pressing the arrow keys which should result in the screen panning. By hovering over and clicking on rooms and characters a context sensitive menu will appear. These will be handled directly by some kind of controller and are dependent on the game state. Clicking on GUI elements will result in "indirect" inputs which will be handled by the GUI scripts in a hierarchical manner. Outputs include graphics and sound.

Data

Physical Memory

The following assets will be compiled into a XNA internal format that allows it to load them at runtime without needing to know a lot of different file formats.

- Art assets
    - Models
    - Textures
        - Model textures
        - UI
    - Shaders
    - Sound
        - Music
        - Effects

The game will be able to save the current state to memory if the user wishes to save his progress. This will consist of player statistics, such as the captain’s current experience, how much money the player has and the number of missions completed.

When the player shuts the game down it will save the current AI states. These will be the different personalities contained within the ship at the time of shut down. It will contain information such as mood and relations with other NPC.

Primary Memory

The game will need to contain a graph of all loaded entities in the current session for it to be able to keep track of their positions in relation to each other.

It will have to keep track of the current game state in order to execute the correct actions at the right time. This will also be used when saving the progress to physical memory.

During set up of the play session the game will have to load the saved AI states into main memory to be able to work with them. It will during the play session modify these states according to changes in the environment and then save them to physical memory.

Code

This section is a general description of each module and its role in the system.

Core

Core is the base of the engine and the class that ties everything together. It's responsible for initiating all the parts of the engine that is needed (the modules mentioned in this chapter). It also creates the links that is needed to interface the game with the Lua scripts that will do a lot of the game logic. This is done using the LuaInterface package. The core simply acts as a hub through which information flows while most of the work is delegated to the more specialized modules.

Scene Graph

The scene graph is responsible for the logical and spatial structuring of the entities in the game. It arranges the entities in a tree structure by using scene nodes to represent either logical boundaries or models. It also contains things like the game camera and a representation of the game time. The scene graph also handles its own updating and drawing by recursively traversing the tree. When drawing it also culls all the nodes that are not present in the cameras view frustum.

The updating and drawing is done by calling those functions of all nodes in the tree. If the tree is a purely logical node the updating and drawing simply moves on to the next node. If it has an entity connected to it its functions will be called in order to draw it on the screen.

Input

To be able to catch inputs the C# function *IsKeyDown(Key)* is passed into Lua. This function works as a wrapper around the XNA input handling. The Lua script then uses this function to check if a button has been pressed and if that is the case it either calls another Lua function or a C# function to execute an action that is specific for that key.

Resource Manager

The resource manager is a class that contains all the necessary data structures that holds the assets of the game. This is done to avoid unnecessary memory overhead by collecting all assets in one place. When an asset is needed the function that needs it simply calls the correct retrieve function with the name of the asset.

Interface

Currently the interface is written entirely in Lua. It consists of a series of tables each representing an interface item. The items have properties such as type, position, size and texture as well as a list of items that sits on top of it. At the moment there is only support for frames and buttons but in the future more elements will be added such as lists and combo boxes.

Message Passing / Event System

This part has not yet undergone development but the key use of it is that it should, instead of specifically calling functions to perform tasks, send a message that specifies what needs to be done. The reason for this is to make the engine more extendable and reusable.

Camera

The camera class contains the camera or cameras used in the game.

The game camera is a third person camera that is attached to a node in the scene. This node is movable and the camera is set to keep itself on a set distance from this node. The camera holds the view- and projection matrices for the game that controls what is being shown and how.

The camera itself handles no input; it simply receives move orders from an external source and then translates or rotates the target accordingly.

AI

All the classes used for the AI.

Actions

Actions have an identifier, a target, duration and a time to live.

Events

Events are used for the appraisal method of the AI. Events can either be Action, Event or Object. Events have a time to live, impact, target and sender.

Agents

Agents store all the information about their mood, personality, needs, memory and other variables used in order to track actions.

Action Trees

Trees are constructed to store all the actions as leafs. All internal nodes on the way down the tree are conditionals.

AI (Generate actions, choose actions, execute actions, appraisal)

Actions are generated by traversing the Action Tree. With all the possible actions found the goal oriented behavior algorithm will pick out one action for the agent. This agent will then execute the action until interrupted or done. After this the appraisal is run.

Path engine

The path engine uses a Path class that represents a number of nodes that the agent can walk along, a graph for storing the ship nodes and a priority queue used in the path finding algorithm. The path finding is done by using an A* implementation.

Operation

User types

The intended users of the game are casual gamers. That is, persons who don't like intense and stressful games. The target audience of the game is people in the ages of 16-30 years old. The users are supposed to play the game like any other game; however the thing that separates our game from many others is that we offer the possibility to how long time the player wants to spend playing a mission. By doing this we offer the player a way of controlling the amount of time he or she sits in front of the computer.

Installation

Normal installation applies, that is an installation like any other software. The system is then configured into the lowest possible settings if several such settings exist. If other settings are wanted, the player changes that as he or she wants.

Licensing

License type not yet decided.

Evaluation

A first "test" has been performed with a wide assortment of student in the Interaction Design master program as testers. The test consisted of showing an early version of the inside of the ship model in order to visualize the view that most of the game will be played through. The concept and base mechanics of the game were then explained. Furthermore, an early version of the scheduling interface (discussed in the "Gameplay" section) was shown and opinions about it were encouraged.

The feedback brought up some interesting points. Many found it hard to understand the overall goal of the game as well as the main reason for playing it. In other words, the reason for why the game would be fun was unclear. We received much valuable input on the interface which led to a complete redesign of some of the elements. The full list of feedback can be found as an Appendix.

All code must be properly commented and read by another person than the one originally writing it. This will hopefully ensure a good average quality of the code as well as other game assets.

Appendices

Low Fidelity Test Feedback

Group A

Very nice simulation of a space ship. You guys have really got started and had nice pictures and seem to have it very clear about how the final game should be our how you want it to be. The only aspect which we can find is that the target group is very narrow/thin in our opinion.

Group B

The control window could be a bit smaller so that the player can see more of the background around it. Even if the window would be a bit smaller then it would be possible to see the content clearly. The window is a little bit transparent, It could be easier to read and see the content if it wasn't transparent. The user understands that it is the temporary window which can be removed because he/she can se the background (under, around the control window).

Group C

Very good-looking low-fi prototype, really got the feeling of how it might work. We like the way that you gain experience and that you get more and better crew and new things. We would like to see that you buy expansions for the ship rather then you buy a whole new ship. Easier to recognize the ship and easier to find the right rooms. We would like to see a clear goal with the game, a good story. Would be cool if the story changes depending on the choices you make. If you decide to be good, you get one story, and if you want to be evil you get another story.

Group D

Sims in Space

The game seems like it potentially would be really fun but the layout needs some thinking through. The menu system might take up a bit too much of the screen real-state, especially if the player needs to spend most of his time in this interface. Pausing the game might not be that good of an idea either when the player is in this menu since the game will take forever to finish then and the player will only get frustrated.

AI: You seem to have a big focus on developing movement and decision making but the actors' actual time spent on moving and acting on decisions seems to be far less and totally disproportionate in comparison to the time they spend just standing around doing their job. This is just the impression we have gotten so far, if they actually move about more then the development effort might not have been in vain.

Group E

About

- 3D game
- Player oversees a spaceship to ensure it runs smoothly

- Personalities are the employees which you as the player hire and fire. These characters unlike in other video games do not come with details on their abilities (e.g. Dexterity: 8/10), instead the player has to observe their actions with other characters to determine if they are doing a good job.

Feedback

- Indirect control over agents is an interesting concept
- Interesting interaction between agents (agents can indirectly influence on each other)
- Although a "more working" (even if more basic - say, 2D, paper, etc.) prototype would have been nice, the 3D structures we were shown let see heavy work is being done on it.
- Making the spaceship modular with replaceable areas will save you time - no need to redo many of the common areas. Just watch out for people's reaction, they might get bored of the common areas. In that case build another ship.

Group F

The background of the community game which involved space and cool spaceships is attractive and interesting. However, the goal of the game seems not so clear. And couldn't find many reasons about why the community should be in a spaceship. It is good if there are more game elements involved space issues for example mining, space war or defend a planet. Also because it is a community game, an online game would make much sense.

Group G

Distinct and well developed concept. Will be interesting to see more of the interaction between characters, and what possible goals and sub-goals that the players will have when using this product. The strive for making the management part of the simulation more realistic by providing input in textual form etc. is a good idea, is it possible to do more in that direction? Like e.g., having the management tasks be performed in-game in some way?

Open Issues

Glossary

Lua - a lightweight, reflective, imperative and functional scripting language

Bibliography

Reference

[OCC]
http://ruebenstrunk.de/emeocomp/4e.HTM#kapitel41

[MASLOW]
http://webspace.ship.edu/cgboer/maslow.html

[OCEAN]
http://www.uwm.edu/~vince/psy205/wwwcourse.205.lec12.fivefactormodel.handout.htm