MIDDLE EAST TECHNICAL UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

CENG 491
SENIOR PROJECT DESIGN

Fall 2004

Final Design Report

WIRTUAL WISION

'Beyond The Sight'

Doğan Sever
Ercan Şahan
Alper Taş
Kubilay Timuçin
Rakıp Yaşar

January 2005

Table of Contents

1. INTRODUCTION

1.1. Purpose of the Document

This document has been prepared to introduce the design details that will be referenced in the implementation phase. The document brings the system to a final design point where all the requirements previously defined in the Requirement Analysis Report are met and the initial design considerations in the Initial Design Report are revised. In this report, all system packages, modules and architectures are documented both verbally and schematically to the group members who will actualize the system in the next semester.

1.2. Scope of the Document

Giving a general overview of the system, the document specifies the design issues related to game scenario, objects, hero and its capabilities. Also, the whole system architecture is explained with functional, behavioral and structural modeling methods. The data being passed between the modules and external storages (files) are discussed. The external file formats are determined and mentioned in detail. The way that the whole system handles the inputs is designed. How state transitions of puzzles and objects in the game are handled is explained. The connection between the GUI and the internal program components are clarified. The auxiliary system component: map builder is considered with respect to the design methodologies. The system is represented using Unified Modeling Languages (UML) diagrams in detail. Functionalities and roles of all components are provided in their related explanations. In particular, the entire system components and the interfaces that interconnect the main components are defined. The two system aspects: artificial intelligence (AI) and game scripting are detailed in terms of their design structures. Lastly, the tools, libraries and the external resources to be benefited from are defined.

1.3. Terms, Acronyms, Abbreviations

3D,	Three-Dimensional
AI,	Artificial Intelligence
CFD,	Control Flow Diagram
COV,	Circle of Vision
CPU,	Central Processing Unit
DFD,	Data Flow Diagram
Esc,	Escape Character
FOV,	Field of Vision
GUI,	Graphical User Interface
LSP,	Last Seen Point
MB,	Mega Bytes
MC,	Mouse Click
MS,	Microsoft
NPC,	Non-Player Character
ODE,	Open Dynamics Engine
OpenGL,	Open Graphics Library
PC,	Personal Computer
SDL,	Simple Direct-Media Layer
STD,	State Transition Diagram
UML,	Unified Modeling Languages

2. SYSTEM OVERVIEW

2.1. System Purpose

The purpose of the software is to provide the PC game players with an easy-to-learn, easy-to-use, hard-to-master, realistic, enjoyable adventure game with arcade components. To develop flexible, modular and easy-to-develop game software is also a crucial target from the architectural perspective. Moreover, one of the goals of the system is to present a new computer game with 3D graphics supported by sophisticated properties like lighting and shadowing.

2.2. System Scope

The software will be an adventure computer game. The name of the game is "Beyond The Sight". It is going to be supported with 3D graphics with rich lighting and shadowing, arcade components, where the players will face "intelligent" enemies. The ability to load 3D scenes to the game will be enabled. On the other hand, the development of the system will be made more efficient using scripts.

The game is based on missions to be completed by the player who will face with several obstacles towards the final target. In particular, the player plays the role of an agent who is assigned to investigate secret information in a foreign country, rescue the hostages and/or destroy the base. The details of the game scenario will be mentioned further in the report.

Several multimedia fragments will be included to present the game and to introduce how the game is played. An auxiliary application, the map builder, will help us generate the default maps delivered with the game software and the ones to be available on Internet. Unlike the ordinary viewing techniques, the game will have isometric camera viewing in which fixed camera views are applied. This technique is very much compatible with the nature of adventure games and also facilitates reducing the calculation of predicted scenes. Besides, the players will be provided with first-person view option to make them able to have better vision.

To provide the modularity of the system, most of the program data will be stored in external files instead of hard-coding them. For example, the model information of objects, the dialogs of hero are kept in such files. This is the convenient way to organize the data that might be subject to change.

To provide an easy management for the installation of the product, an installer file (install.exe) will be generated.

3. DESIGN CONSIDERATIONS

This section describes many of the issues that need to be addressed or resolved before attempting to devise a complete design solution.

3.1. General Constraints and Goals

The general constraints in the design phase are devoted to performance and end-user environment worries. The development environment is therefore chosen accordingly. The operating system is Windows and the coding tool is Visual Studio. C++ is the programming language. The 3D graphics will be implemented with the help of OpenGL. The development environment analysis was performed in Requirement Analysis Report. In the following sections in this report, the other tools, libraries and external resources are considered from the point of design view.

To obtain a satisfactory game performance, the software should be installed on a PC with a certain configuration. Fortunately, these specifications are mostly satisfied by the current PCs in the market.

There are several points to be considered. Efficiency is one of the most important of them in the design of this system since game is real-time software for which quick responses are vital. Therefore, it is the fundamental purpose to develop a more efficient system in terms of speed and performance.

One of the main solutions to achieve this purpose is not to define most of the game world objects in the physical engine. The basic reason behind this is that the physical engine, ODE in this case, consumes the system resources extremely if all objects are defined to have physical features like mass or bounding volume. Instead, the objects except for the main character and the enemy characters will only be implemented as graphical entities. Additionally, the ground, and fixed entities like walls will be created in ODE to define the surrounding environment physically.

Secondly, it is a fact that disk operations are the slowest compared with the other processes like memory read/write or CPU operations. As a result, disk read/writes must be minimized to provide the efficiency. Ultimately, the data stored in files are read only at the points where the player is not inside the game screens.

Another approach is to develop an easy-to-use game as much as possible. For example, to move the main character by using keyboard seems to be difficult for the players. To overcome this problem, pointing the destinations using mouse clicks is thought to be more reasonable to serve the ease of movement control.

In brief, efficiency and usability are the two important requirements to be met and the primary goals to be reached determining the basic guidelines of design considerations.

3.2. Development Tools and External Resources

The software will be developed on Windows XP Operating System using C++. Microsoft Visual Studio will be the tool for compiling, executing and debugging the C++ codes. In order to create 3D graphical environments, OpenGL will be used. Besides, multimedia content will be created using auxiliary programs like Adobe Photoshop.

To handle the physical operations in the game environment, the physical engine Open Dynamics Engine (ODE) will be used. With the help of ODE, the game space, world, ground and other fixed entities will be defined. Also the main character will have physical existence in the program to detect collisions of him with the corresponding world entities.

The object models to be used in the game environment will be prepared on 3DS Max 6. These models will either be taken from Internet and modified using this program or be created from the scratch.

The methods of Simple DirectMedia Layer (SDL) will control the audio data. These audio data will be generated using sound editors.

Scripting language Python will ease the modification of the internal data of the system. Model/object properties and their states are among the internal data that need modification during development and testing of the software. For this purpose, Python 2.4 interpreter will be used. Another development tool will be used for providing the communication between the high-level scripting language Python and C++ codes. This tool is SWIG. SWIG is a code development tool that makes it possible to quickly build powerful scripting language interfaces to C or C++ programs.

Software engineering tools help us create design documents. The time schedule for the development cycle is generated using MS Visio and MS Project. All necessary diagrams (i.e. DFD, CFD, STD, etc.) are drawn using Smart Draw. A UML tool (Rational Rose) is useful to draw the class diagrams.

To ease the installation of the final product, an installer file will be generated using an install wizard creator. Candidates are Ghost Installer and Installer Creator.

For the documentation of the project reports, MS Office and Adobe Acrobat Professional are used.

3.3. Development Methodologies

A Linear Sequential Model is being followed within the development life cycle of this software. This model "suggests a systematic, sequential approach to software development that begins at the system level and progresses through analysis, design, coding, testing, and support." [1]

Additionally, a prototype implementation takes place within the life cycle of the system. The prototype helps us see the probable problems that can be encountered in the implementation phase. Since the prototype has the basic features of the system, there is the chance to see the big picture of the project. This report completes the design phase; and after prototyping, implementation of the project will start in the next semester.

An object-oriented approach is applied in the project development progress. The system is intended to have a high modularity that increases the adaptability and reusability whereas decreases the complexity of the system. Also object-oriented programming provides several concepts and features to make the creation and use of objects easier and more flexible.

4. SYSTEM ARCHITECTURE

This section provides a high-level overview of how the functionalities and responsibilities of the system were partitioned and then assigned to subsystems or components. The main purpose here is to gain a general understanding of how and why the system is decomposed, and how the individual parts work together to provide the desired functionality.

4.1. Subsystem Architecture

4.1.1. Functional Architecture

In this section, the system is represented with a Data Flow Diagram (DFD), which "depicts information flow and the transforms that are applied as data move from input to output" [2]. DFD of the system is composed of three levels in each of which the system is explained in different levels of abstraction. In the first level (See Figure 4.1 - DFD Level 0), the system is represented in an input-output manner. The two input resources to the main system are the keyboard and the mouse, controlled by the player. The two outputs of the main system are the monitor and the speaker.

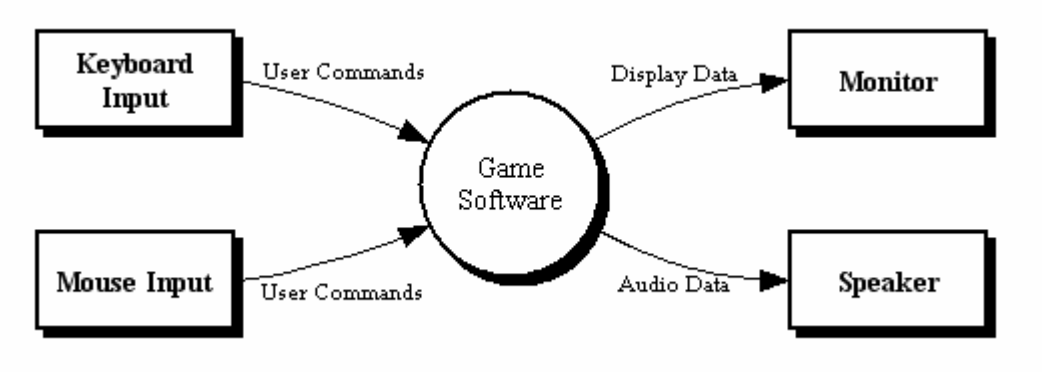


Figure 4.1 – DFD Level 0

User issues the commands to the system via the input resources and gets the system response, which are visual and audio data in this case, from the monitor and the speaker, respectively.

DFD Level 1 (See Figure 4.2) focuses on the main system, named Game Software in DFD Level 0. The system is composed of seven modules and the data flows among them. Also, the files containing external data on the disk participate to the data flow. The seven modules are: Physics Engine, Event Listener & Handler, AI Engine, Graphics Engine, Sound Generator, Data Pre-Processor and Game Engine.

The whole game environment that accommodates all the interactive and non-interactive objects is contained in the game engine. This module also manages the global variables of the system. This module directly accepts the user inputs. Interacting with all the other modules, game engine plays a central role.

As the name implies, event listener & handler listens and handles the internal events that occur within the game flow with the impact of the internal interactive objects. It reacts to the game engine informing the necessary actions to be performed. When a sound is produced, for example, an event is automatically created and the event listener & handler unit handles the event by informing the AI engine to take action.

Who manages the decision-making process of the "intelligent" non-player characters is the AI engine. The game engine informs it about the environmental changes and the AI engine determines the actions of the intelligent characters.

Data pre-processor manages the load/save operations. It is also responsible for loading a new map. To do these, it reads the files on the disk, fills necessary data structures and transmits these data to the game engine. On the other hand, it writes the game save data to files. It is also the data pre-processor's duty to handle all file (except sound files) read/write operations of the whole system.

Graphics engine manipulates and transmits the visual data that it receives from the central game engine to the monitor.

Sound generator serves the requests of game engine and transmits the sound read directly from file to the speakers.

Physics engine is responsible for detecting the collisions of the hero and NPCs with the ground, walls and each other. The other objects are not going to exists in the physical manner; therefore the movements of the characters into these objects will not be detected by the physical engine but be restricted by the game engine who already knows the coordinates that are reserved.

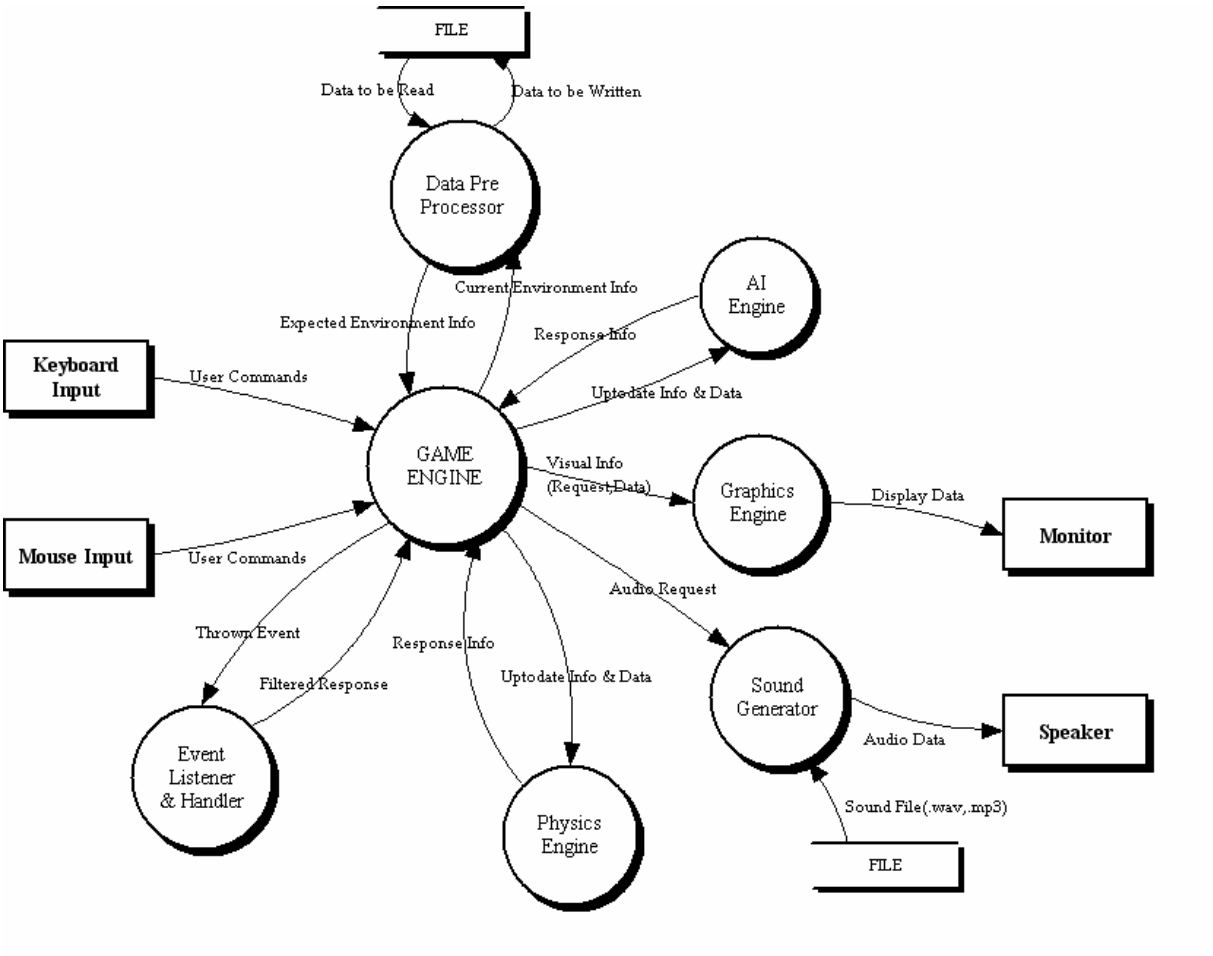


Figure 4.2 – DFD Level 1

In DFD Level 2 (See Figure 4.3), inner structures and the data flows of the game engine and data pre-processor are the two considerations.

Setup manager deals with the whole configuration of the game managing the changes in the graphical, audio and control configurations.

What the scripting module is responsible for is to access and/or alter the internal data. This data includes member variables of object instances or any global variables. Also, it is possible to call any method or function of the program via this module. Throughout the development, scripting module will provide us with the opportunity to manage the internal data of the system without recompiling the source code or restarting the process.

As a whole, these diagrams give an idea of the sub-structures of the whole system and the data flows in between them.

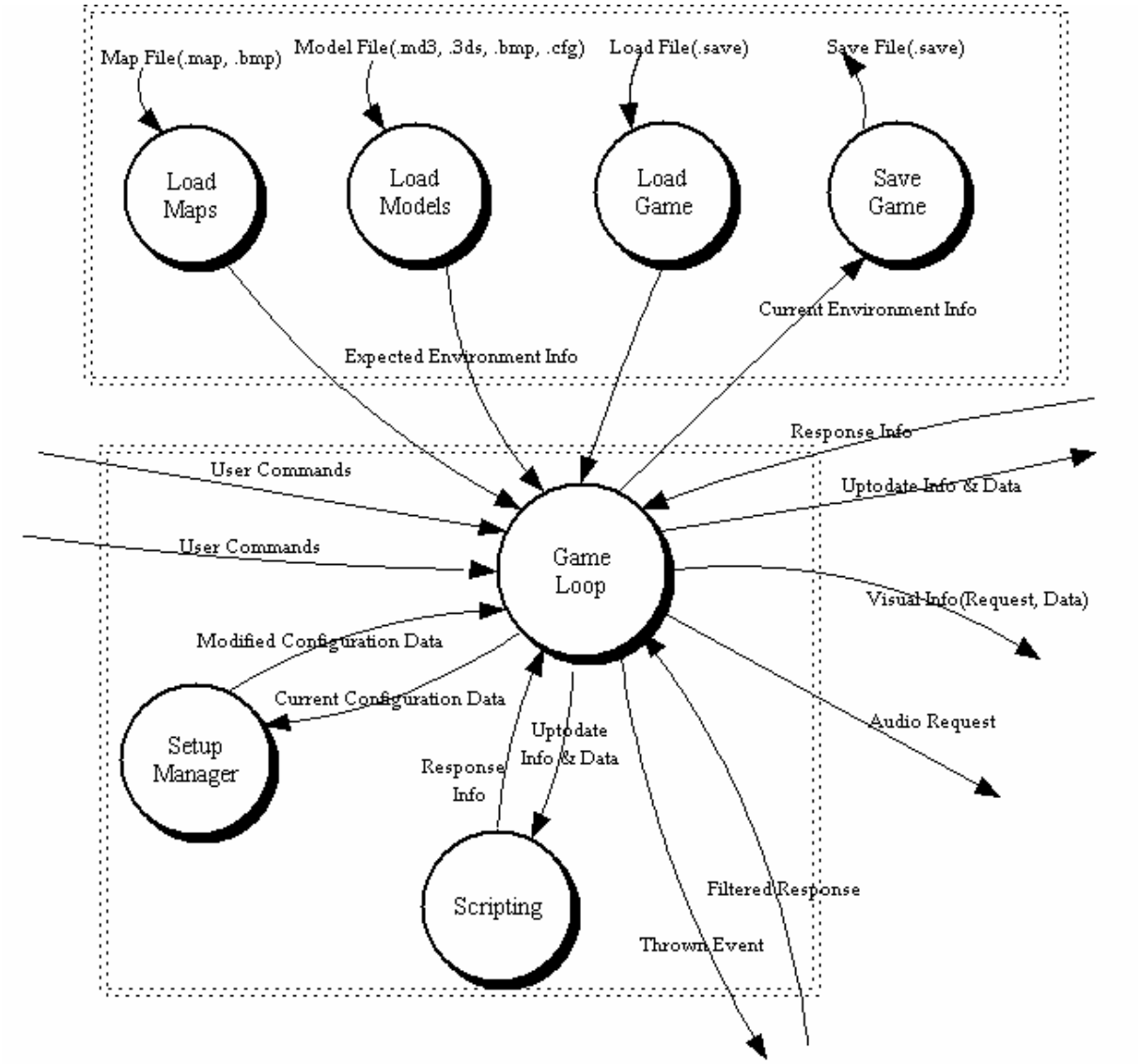


Figure 4.3 - DFD Level 2 / data pre-processor (above), game engine (below)

4.1.2. Behavioral Architecture

To model the reactions of the software to the external events, the State Transition Diagram (STD) is used. In this modeling technique, the STD "represents the behavior of a system by depicting its states and the events that cause the system to change the state. In addition, this STD indicates what actions (e.g. process activation) are taken as a consequence of a particular event." [3]

The STD of the software to be developed (See Figure 4.4) indicates the states and transitions among them. The menu interfaces and the game loop are the two main parts.

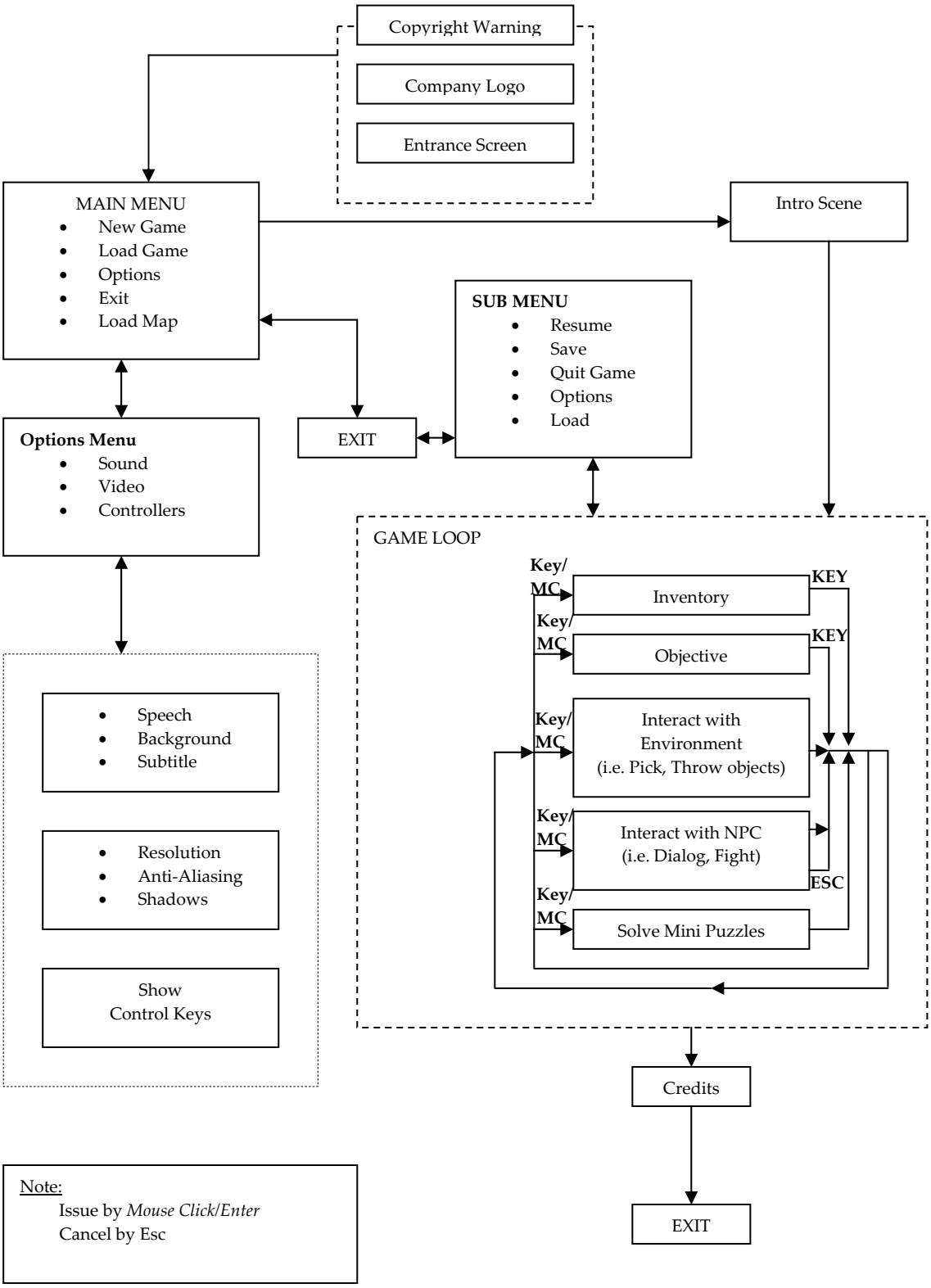


Figure 4.4 – State Transition Diagram

As can be seen in the diagram, Esc (Escape character) is generally used for return to the previous state. Enter and the other corresponding key characters are used for submitting and going one step further in the system flow.

The following three figures summarize the detailed in-game state transitions.

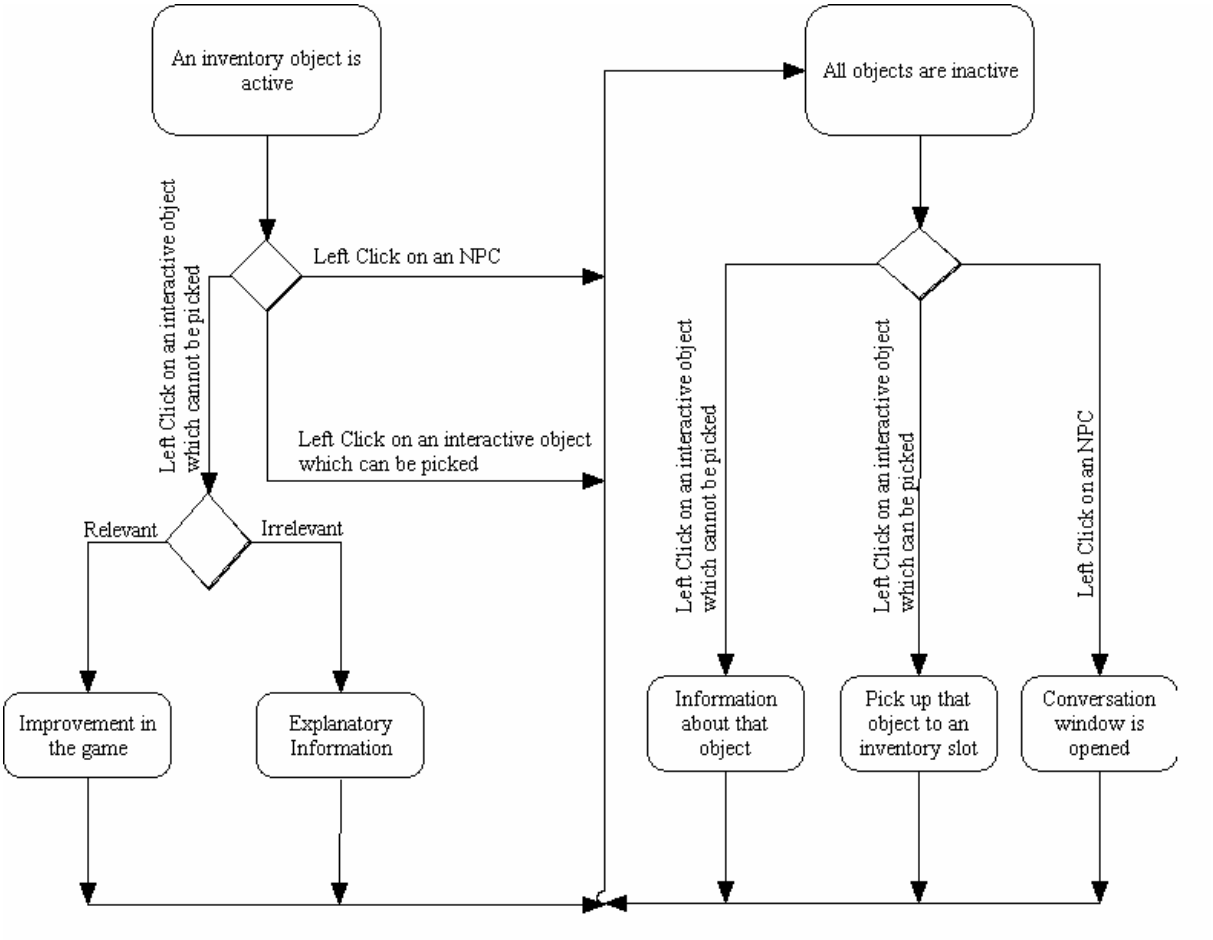


Figure 4.5 – General In-Game State Transitions

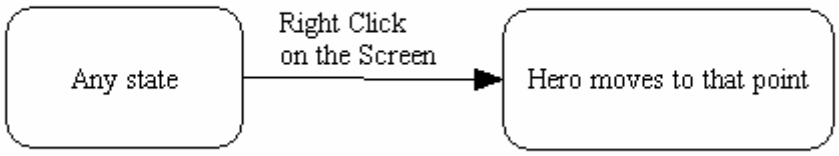


Figure 4.6 – Mouse Right-Click Action

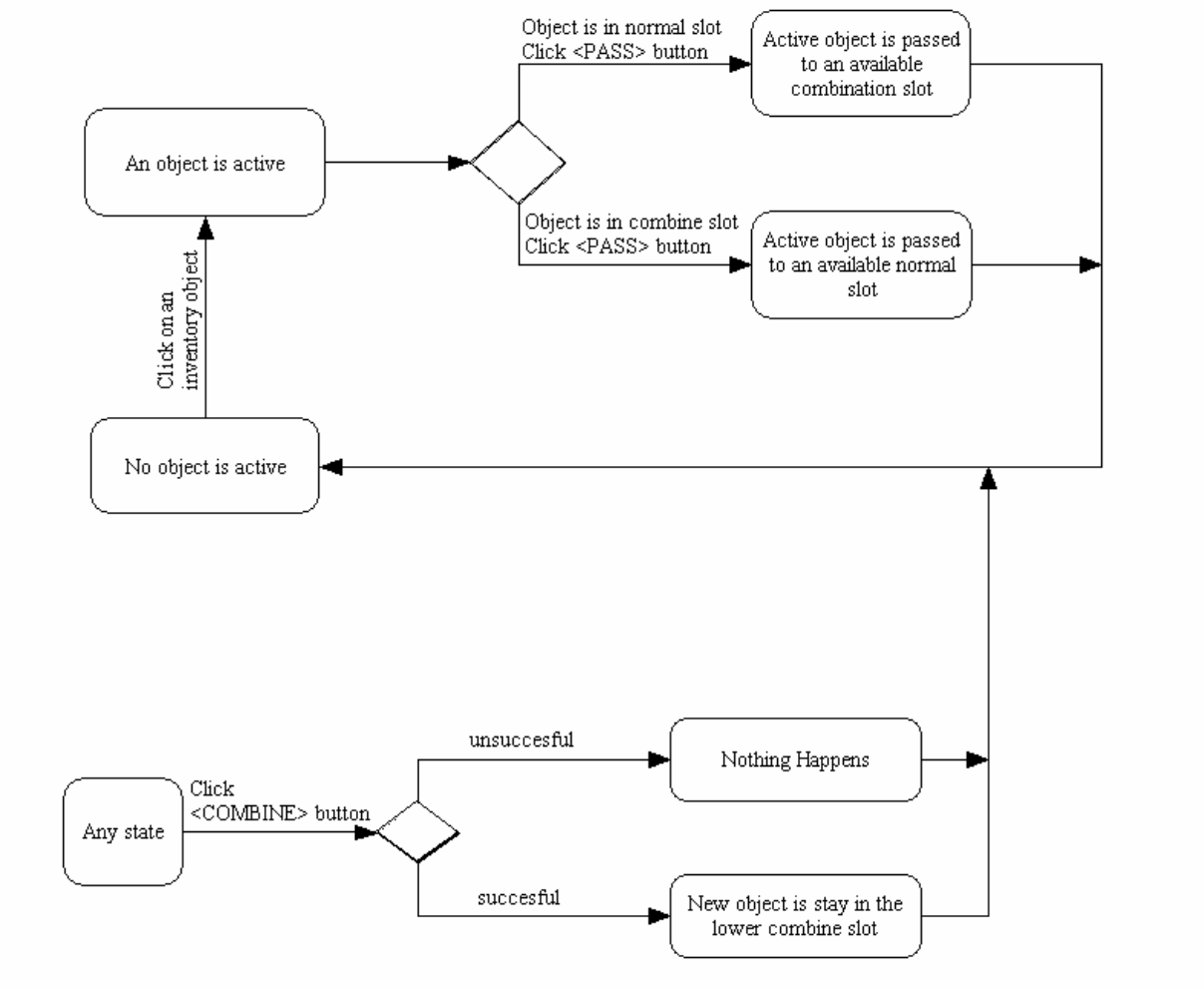


Figure 4.7 – General In-Game State Transitions

4.1.3. Structural Architecture

The data flow among the different modules of the system is displayed in the DFD. These flows are controlled by several mechanisms that assure the consistency and integrity of data flowing from one module to another. The data flows across the modules of the game to be developed are also controlled in some cases. These controls are demonstrated in the Control Flow Diagram (CFD) in Figure 4.8. This CFD is a dual of DFD level 1 in that the modules are represented in the same level of abstraction.

As seen in the diagram, there are three control structures. Firstly, there is a control in the flow of graphical data from game engine module to graphics engine. This control corresponds to the activation of some of the graphical features. For example, through the corresponding configuration menus provided to him/her, the player can optionally disable shadowing. The other control mechanism is the sound control. The flow of audio data is checked against the player's decision on whether to switch on or off the game sounds. Lastly, as stated earlier, saving a game is enabled in only certain phases of the game flow. For this reason, there must be a control point to check if the current phase allows a game save.

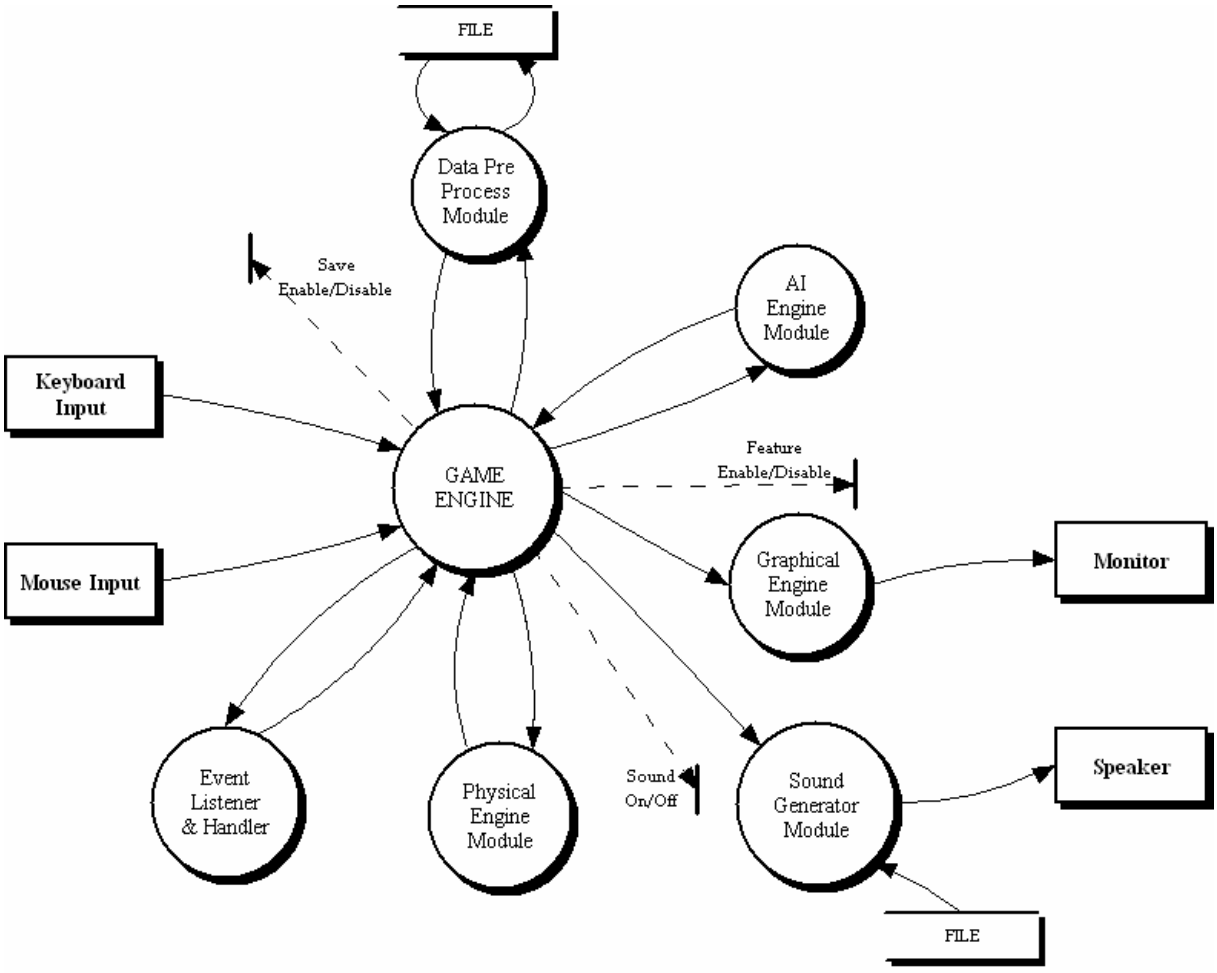


Figure 4.8 - CFD Level 1

4.2. Data Description

This section is devoted to the descriptions of the data stored in files and to be imported at the start of the game and each level. Also, the external data needed during the game flow is described.

There are basically nine groups of data externally stored: maps, models, saved games, pictures, textures, sounds, ODE libraries, SDL libraries and Python files.

A map file has an extension ".map" and keeps the data of all levels that form a single mission. Each level data is organized in such a way that it contains all the data needed to define an object in terms of the amount, position, properties, states and status. A map file is the output of the map builder that is to be implemented as an auxiliary application. The format for a map file is as follows:

```
//** MAP DATA **

map_identifier
number_of_levels

        //** LEVEL DATA[number_of_levels] **

        level_identifier
        dimensions
        number_of_cameras, camera_positions[number_of_cameras]
        camera_areas
```

```
        number_of_spaces, relations_of_spaces[(number_of_spaces - 1)]

                number_of_geometric_objects

                //** GEOMETRIC OBJECT ATTRIBUTES[number_of_geometric_objects] **

                space_id, dimensions, position_vector, rotation_vector, rotation_angle
                object_type, object_identifier, object_id
                number_of_states, state_identifiers[number_of_states], current_state
                number_of_transitions, transition_rules[number_of_transitions]
                number_of_relations, related_object_ids[number_of_relations]

                number_of_physical_objects

                //** PHYSICAL OBJECT ATTRIBUTES[number_of_physical_objects] **

                space_id, mass, density
                dimensions, position_vector, rotation_vector, rotation_angle
                object_type, object_identifier, object_id
                number_of_states, state_identifiers[number_of_states], current_state
                number_of_transitions, transition_rules[number_of_transitions]
                number_of_relations, related_object_ids[number_of_relations]
                number_of_speechs, sound_sources[number_of_speechs],
                speech_text_sources[number_of_speechs]
```

The model files have the extensions ".ase, .3ds" and/or ".md3". These files are adapted from the models found on Internet. 3DS Max 6 is going to be used to differentiate the models. Under the same directory of each model, there will be files with the name "states.cfg" storing the state transition rules of the model. This file is very simply structured.

Saved games are also stored in files of extension ".save". These files fundamentally store the map info that the saved game belongs to and the current environment and state status. The files are output when player selects the game save option and imported when he/she selects the game load option. The format of the saved game file is as follows:

```
//** SAVED GAME **

map_identifier
level_identifier

        number_of_geometric_models

        //** GEOMETRIC OBJECT ATTRIBUTES[number_of_geometric_objects] **

        rotation_vector, rotation_angle, position_vector, current_state

        number_of_physical_models

        //** PHYSICAL OBJECT ATTRIBUTES[number_of_physical_objects] **

        rotation_vector, rotation_angle, position_vector, current_state

alarm_status
sound_position
```

Pictures are the multimedia content to be screened as the introductory scene and in between the stages and levels. The extensions of pictures may either be ".bmp" or ".jpeg". When the pictures are being displayed, related sound and text content are also screened.

Textures have ".bmp" extensions. These files are loaded when the map and/or model is loaded into the game. They are the skins of all the environment, objects and characters.

Sound files are of three types: speeches, music, object sounds. Speeches belong to the dialogs between the hero and the non-player characters. They are to be played with the corresponding dialog text. The extension of a speech file is ".wav". Music files (".mp3" and/or ".wav") are the background music played within the game flow. Thirdly, the objects sounds are the sounds that objects can make, for example, the gunshot and magazine sounds. All sound files are accessed during the game flow when necessary.

Another data file is "globals.cfg" which contains the previously loaded map's name, non-default user control input keys, and the user defined configuration options in the setup menu.

Finally, the rest are the external libraries containing the ODE and SDL methods. Also the Python script files of extension ".py" will be stored on the disk.

5. DETAILED SYSTEM DESIGN

5.1. Unified Modeling Languages (UML) Part

In this section, the system is described using UML methods. The diagrams are the use case diagram, the class diagram and the sequential diagram. These diagrams make it easier to comprehend the system's overall behavior and structure.

5.1.1. Use Case Diagram

The following figure (Figure 5.1) is the use case diagram of the game. All transitions from start to end are depicted in a top-to-bottom manner.

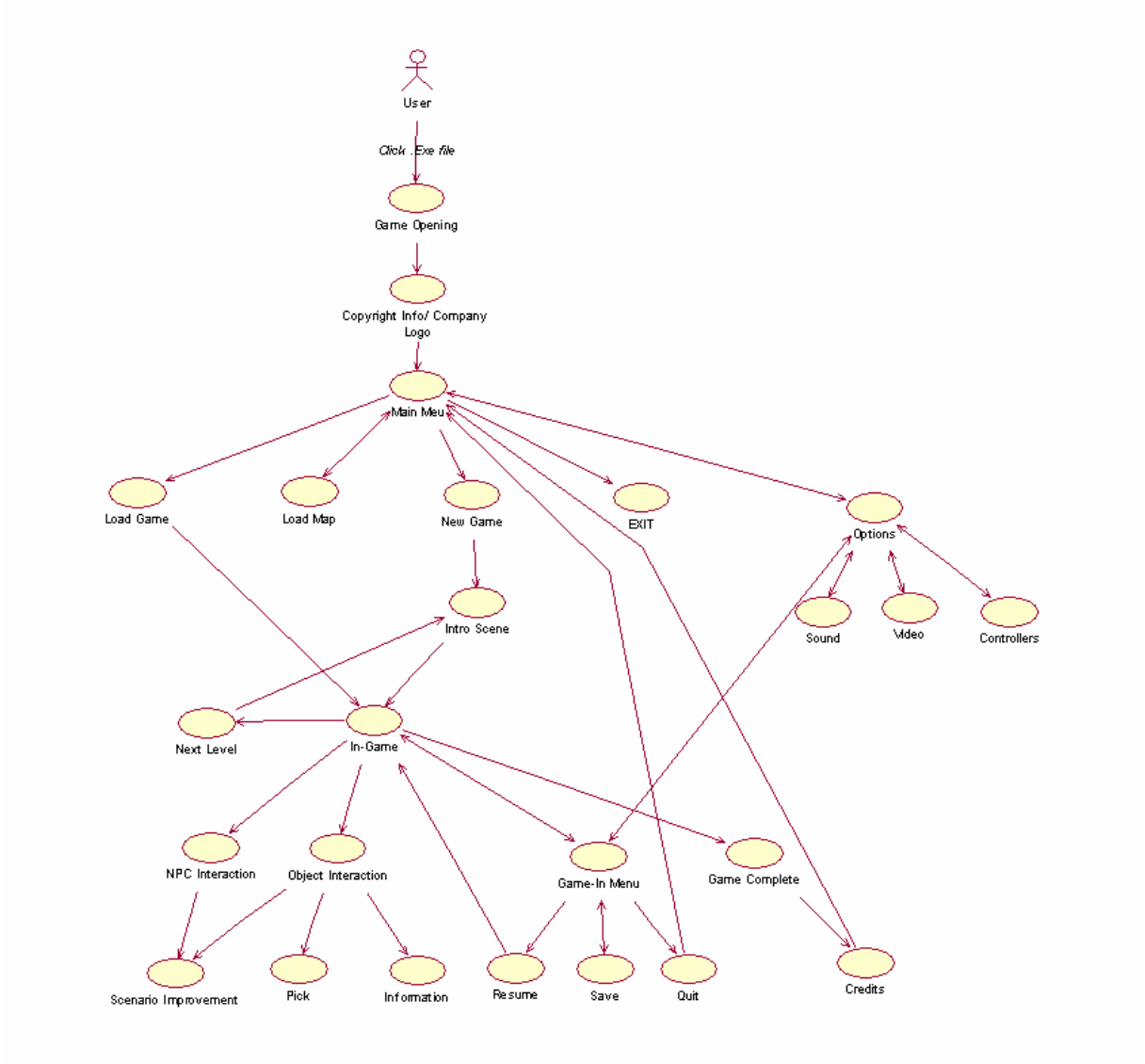


Figure 5.1 – Use Case Diagram

5.1.2. Class-Based System Description

This section describes the classes and their methods in a start to finish manner. It will be helpful to follow the class diagram depicted in Figure 5.2 to understand the system details when reading this section.

The main() program starts with creating a gameWindow (instance of GameWindow class), which holds the game, aIManager, render and eventLH (instances of the Game, AIManager, Render and EventLH classes, respectively). First of all, gameWindow.create() method will be called for creating the main window of the game. With the help of the gameWindow.mainLoop() method, the continuity of the game will be provided. In this method, the methods game.operate(), aIManager.decideBehavior(), render.display() and eventLH.handleEvent() are called in a loop for the corresponding processes in the given order. The method gameWindow.callback() is used for handling the inputs (keyboard and mouse) from the player. This method uses the macros in the "windows.h" header file for this purpose.

To manage the game states some global variables are kept in a header file (globals.h). Game.operateGame() and aIManager.decideBehavior() methods do nothing at the start of the game under the control of the global variables. render.display(), however, calls intro.displayIntro() of its private member intro (instance of Introduction class). intro.displayIntro() displays the copyRightWarning and companyLogo, the two private members of intro, the first of which is the copyright warning string and the second is the path for the company logo picture. Then, render.display() calls render.drawMenu() method who draws the main menu.

When the new game button of menu is pressed, game.menu.newGame() method is called. This method calls game->map.loadMap() who loads the default map specified in the globals.h if it is the first time the player starts the game and has not loaded another map before. The previously loaded map is loaded otherwise. This information is stored in an external file globals.cfg in such a case. The map is read from map file. game->map.loadMap() calls map's currentLevel.loadLevel(). This method sets the scenes (instance of Scene class) array of map's currentLevel, which are the cut-scenes to be displayed in between the stages and levels. It also calls game->models[].loadModel() method to set the models (instance of BaseModel) array of game. The method currentLevel.loadLevel() then initiates the method game->aIManager.getIntelligentModels() who sets the private member aIManager.models[], accordingly. After this stage, the system goes to the game mode. The system modes are flagged by the global variables initiated by the globals.h file.

When the load game button of menu is pressed, game.menu.loadGame() method is called. Menu has a private member, list (instance of List class). loadGame() method calls list.fillList() method. This method fills the list of menu with the names of the previously saved games read from the disk under the save directory. When one of the games is selected and submitted, game->importFromFile() method is called with the selected saved game file name. This method loads the corresponding map and level in a similar manner mentioned in the preceding paragraph. Then, it sets all variables of necessary objects according to the values read from saved game file. It also sets the global control variables accordingly.

When the save game button of menu is pressed, the list of saved games are displayed as in load game, and the player is asked for a save game name. Player may either choose a name from the list to overwrite the saved game or specify a new name if there are still free slots in the list. This is the restriction that only 5 different games can be saved. When the process is submitted, game->exportToFile() method is initiated, who dumps all the current game states into a save game file.

When the load map button of menu is pressed, game.menu.loadMap() method is called. This method calls list.fillList() method. This method fills the list of menu with the names of the available maps read from the disk under the map directory. When one of the maps is selected and submitted, the global default map variable is changed with the selected map name. Then display returns to the main menu.
When the configure options button is clicked, global control variables are changed to inform render.display() to draw the configuration menu. The user configures sound and display options then submits; at this point the corresponding preferences are stored back to the members of game->menu.config. Initially, the configuration variables are set from the globals.h.

When exit button of menu is pressed, game.menu.exit() method is called. This method first calls game.menu.config.saveConfig() who saves the user preferences to globals.cfg file. Then it exits the game window terminating all running processes of the game.

When the game starts, callback() method directs all inputs to corresponding actions under the control of the global variables that determines the game states. The actions to be taken are passed to game.operate() and this method operates the game flow accordingly.

If the inputs are related with the actions or motions of the main character, then this method calls necessary methods of the main character model residing in the models array of the game. These methods may be findPath(), changeDirection() or goTo(). Since hero is also a model having states, the render needs to consider the state of the hero to draw him in either walking, running or crouching etc.

If the mouse is clicked on a item in the inventory screen and this item is an inventory item, then the activeInventory (a private integer member of game) is set indicating the index of inventory array (instance of BaseModel class). There are three buttons on the inventory screen: pass, combine, objectives. When combine is pressed, two inventory items passed –by pressing the item and then pass button- previously form a new inventory item with new functionalities. This is achieved by creating this new item in the inventory array of game. In fact, this item is a pre-defined model.

If the input is keyboard Esc character, then the global mode variables are set to menu mode to inform the render to display the menu and stop game.operate and aIManager.decideBehavior().

When we look at the AIManager class, we see that it copes with the events related to “intelligence” issues. It holds information about intelligence assigned models. The core method of the class is decideBehavior(). This method is responsible for intelligent models’ actions in case of events which needs the model to act according to decisions.

Another class at this level is Render class. As the name implies this class is designed for rendering the game. It consults “globals.h” file, which contains initial global values and display modes, and according to display mode value it reads from that file, Render class’ display method is activated. Render class has changeCamera() method, drawModel() and drawMenu() methods.

EventLH class is responsible for event handling in the game flow. It collects internal events in an Event class type array, and according to the arrival times for the events, calls handleEvent() method. That method informs necessary objects about the event.

Event class keeps the sender object’s pointer as a BaseModel pointer, the occurrence position of the event as a Point type and the type of the event as a string.

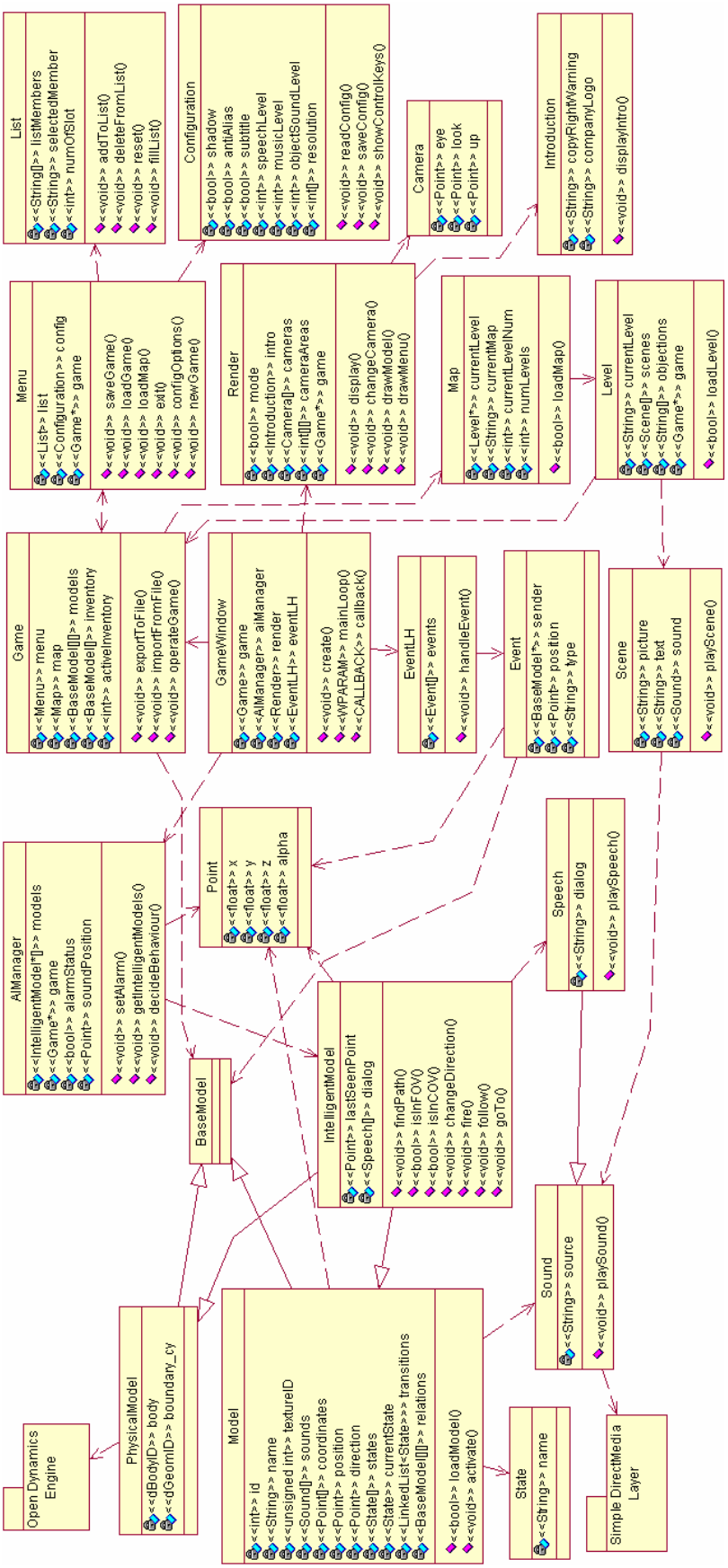


Figure 5.2 – Class Diagram

We have BaseModel class. We have it since we want to treat all different kinds of models as the same. These different kinds of models are PhysicalModel, IntelligentModel, and Model. PhysicalModel is given for models that are supposed to interact with ODE. IntelligentModel does inherit from PhysicalModel class for physical properties. That class has methods for intelligence checks like findPath() and follow(). IntelligentModel is also inherits from Model class for that class' features. Model class does hold information like position, current state. Model class has loadModel() method.

Shortly saying, Model class is used for holding models' default necessary information and loading the model. Model class has activate() and trigger() methods for state transitions between Model instances. A Model instance knows its related Model ids, and according to these ids, if a model is activated then that model's trigger() method will be called. That method calls activate() method of each related Model whose id that we are aware of. Corresponding activate() methods are responsible for these activated model's state transition.

Sound class is another class in the development. Class has playSound() method in order to supply sound support in the game. That class holds string name for sound to play.

One other class is Speech class. Like Sound class, Speech class has string for dialog sentences. Class's playSpeech() method does display dialogs on the screen.

State class is another class of the game. It holds names of states basically.

Point class is just for holding the vector coordinate in space as x, y, z.

List class is used in the menu object for listing the loadable saved game data and the loadable maps data by the player. If the player select one of the object in the list, then the list object send the selected one to the menu object.

Moreover, Configuration class is called from the menu object, and it holds the current game configuration data like shadow, antiAliasing, subtitle on/off, musicLevel and so on…

Introduction class is related with the render object and it is called by the render object at the beginning of the game.

In the map object, there is a Level class type object which holds the current level information as name of the level, and available scenes in that level. Scenes are in the type of the Scene class. Level class also contains a method named as loadLevel() for reading the level data from the file.

As mentioned above, Scene class type of objects hold the pictures, texts and the sounds of the some temporary game scenes for showing them in the game flow.

5.1.3. Sequential Diagram

Here, the order of the actions processed in the game loop is represented in the following figure (Figure 5.3.a and 5.3b).

Figure 5.3 – Sequential Diagram (a)

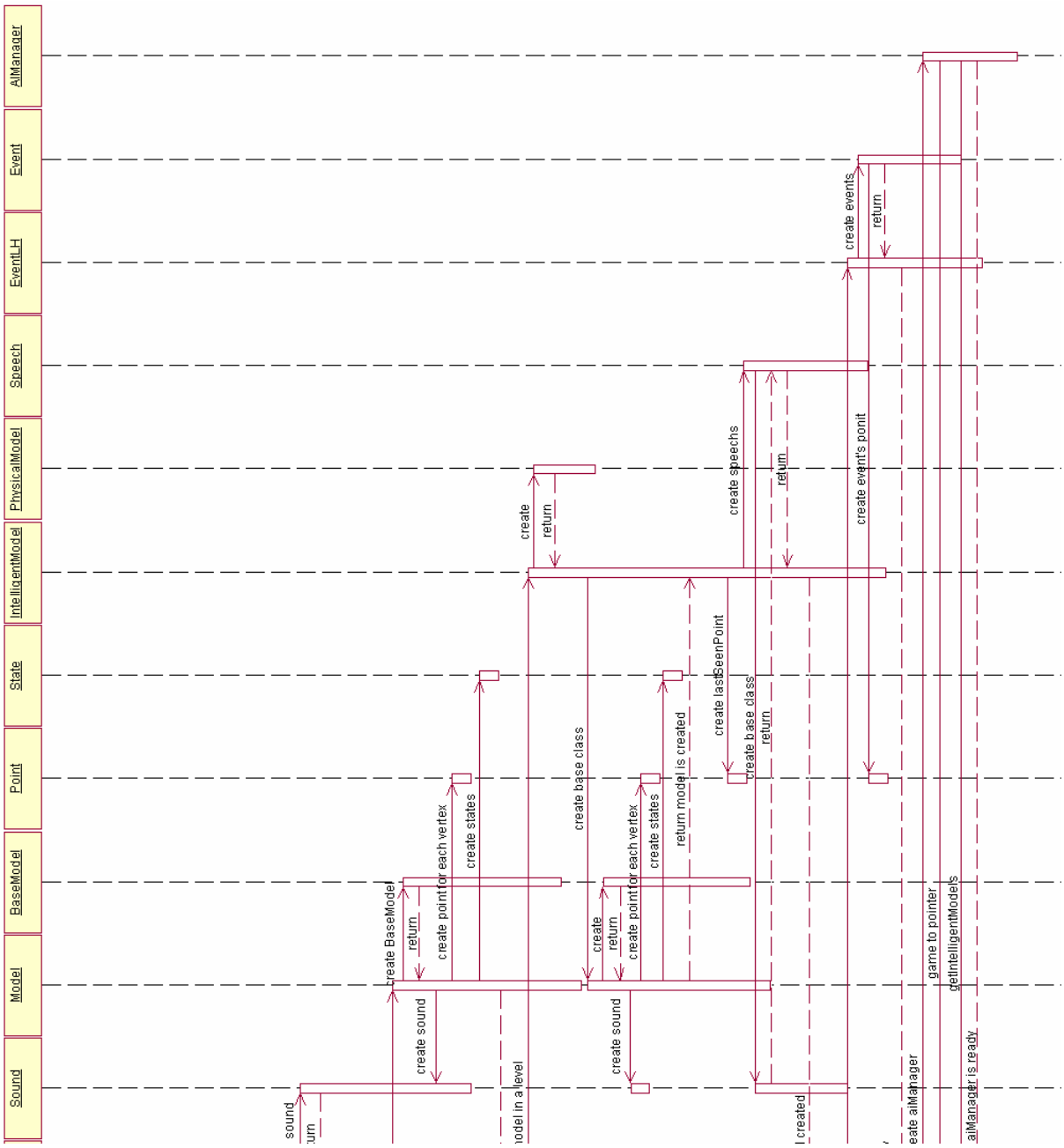


Figure 5.3 – Sequential Diagram (b)

A similar sequential representation is given in the Figure 5.4 for the inner game actions.

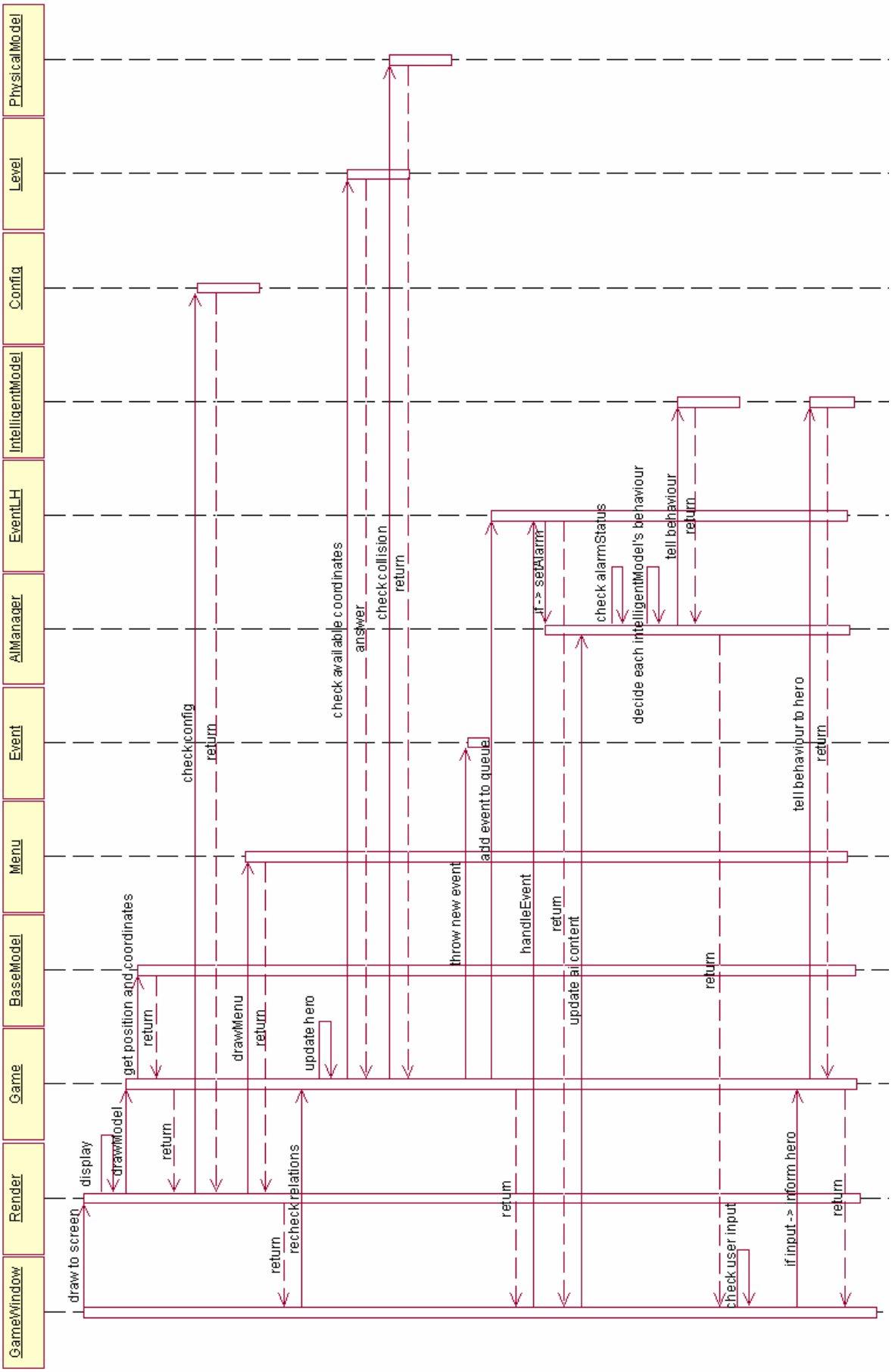


Figure 5.4 – Sequential Diagram

5.2. Additional System Aspects

5.2.1. Artificial Intelligence Concept

Implementing "intelligence" on some of the game characters is one of the main goals behind of which there is the motivation that a game with "intelligent" enemies makes it more challenging and realistic for the players to toy with.

What is meant by "intelligence" in particular is that the enemy characters in the arcade scenes will be able to recognize the hero and follow him until they defeat or be defeated. They will also be capable of realizing a danger and taking precautions when an extraordinary sound (i.e. gun shot) is heard.

In this section the algorithms providing these features are detailed. They are basically about how the field of vision (FOV) of the enemy characters changes, how they follow the hero and behave in general and how they react to an extraordinary sound.

Field of Vision (FOV) Algorithm

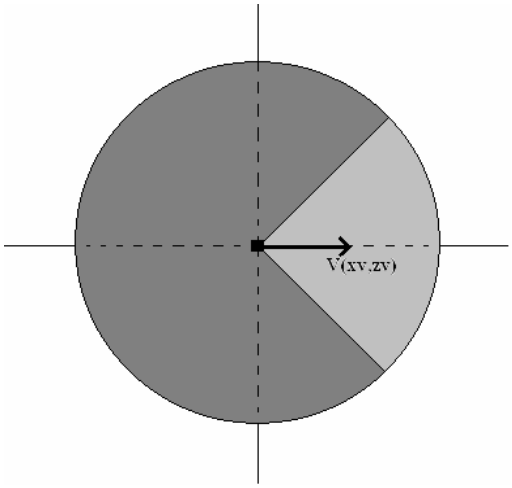


Figure 5.5 – COV and FOV

Check if the hero is inside the circle of vision (COV) of the enemy, (See Figure 5.5)

If so, check if the hero is inside the field of vision (a slice of circle of vision) of the enemy.

If so, turn toward the hero's position and start firing.

The look direction of the enemy character is the position vector from its position to hero's position.

Behavior Algorithm

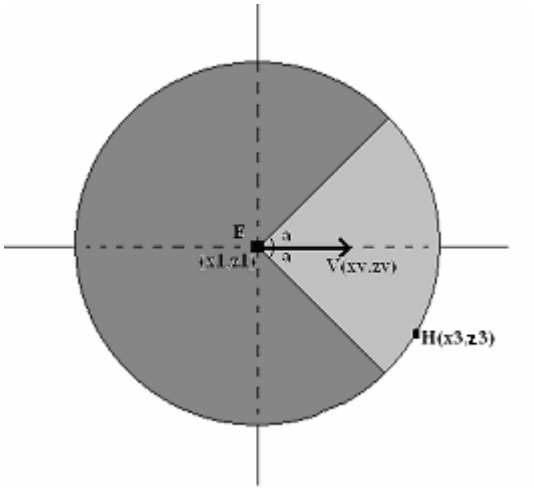


Figure 5.6 – Last Seen Point

<u>Follow/Pursuit</u>

The point where the hero gets out of enemy's FOV is defined as the Last Seen Point (LSP). It is shown as H(x_3, z_3) in the Figure 5.6. E(x1, z1) is the point where the enemy stands.

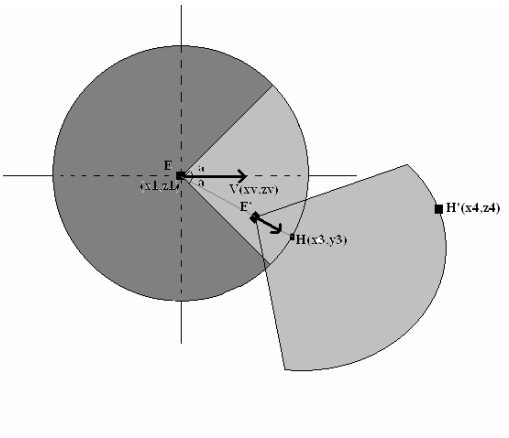


Figure 5.7 – Pursuit

When the hero gets out of enemy's FOV, the enemy stops firing and starts a movement towards the LSP. This movement action is preemptive, that is, if the hero falls into the FOV of enemy at some point H'(x4, z4) during this action, the enemy starts a new movement action towards H' (See Figure 5.7).

If enemy completes its movement action to the LSP with no interrupt, then waits for a while, looks around with a complete rotation. If enemy fails to see the hero, then starts a movement action towards its original location.

Enemy uses path-finding algorithm in movement actions.

Fire

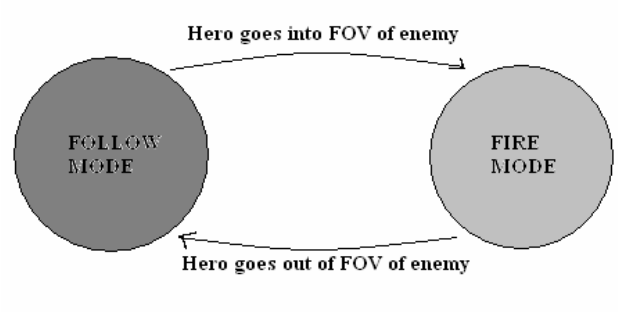


Figure 5.8 – Modes

If hero comes into FOV of enemy, enemy goes into FIRE MODE (See Figure 5.8). Fire direction is the vector from enemy's location to hero's location.

Enemy has a hit/miss rate for successful shooting. This is determined with its distance from the hero. When the enemy is closer to the hero, its miss/hit rate decreases/increases.

Hit Rate + Miss Rate = 1.

When the hero somehow starts firing without silencer, the gun produces a sound that can be heard by all the enemies. Then the enemies fall into the alarm state. Some starts a movement action towards the point where the sound is produced and others keep their positions to guard the gates, for example. Enemies use the path finding algorithm also here.

5.2.2. Scripting

Using an existing scripting language saves the time and cost of developing a custom language. Python is an excellent choice for a game scripting language because it is powerful, easily embedded, has most of the advanced features that make scripting languages worthwhile, and can also be used for automating production.

First, some interface files will be developed that contains the declarations of which classes, variables or functions to be accessed or altered in run-time. These interface files have extension ".i" and are used by SWIG to generate the Python scripts that are ready to communicate with the running application.
At this point, it is worth mentioning where these scripts will be applied, in 'Beyond The Sight'. As mentioned earlier, to ease the development phase, some run-time changes in the internal states of game data is required. There are several points where scripting will help us.

First of all, testing the state transitions and object relations is a bothersome task. To do effective testing on this, it would be very feasible to access the related methods and variables of the object models in the run-time. Therefore, the interfaces to be developed will include external access to the methods of models that manage the states and the transitions of the objects.

Secondly, to test the camera views in the game, scripts will be used for changing the active camera.

Furthermore, it is an important part of the development to check if the game flow reacts appropriately to the internal events. To do more effective testing on events and the reactions of the system, scripts will be used for creating new events in the event listener and handler module.

Consequently, any time we need to test an internal data and game flow, scripts can be generated using interfaces handled by SWIG and the Python interpreter.

6. AN AUXILIARY COMPONENT: MAP BUILDER

This section discusses the data flow between the modules of the map builder and schematizes the software with its class diagrams.

Figure 6.1 focuses on the main structure of the map builder. It depicts the DFD Level 0. The software has two inputs, keyboard and mouse; and one output, display monitor.

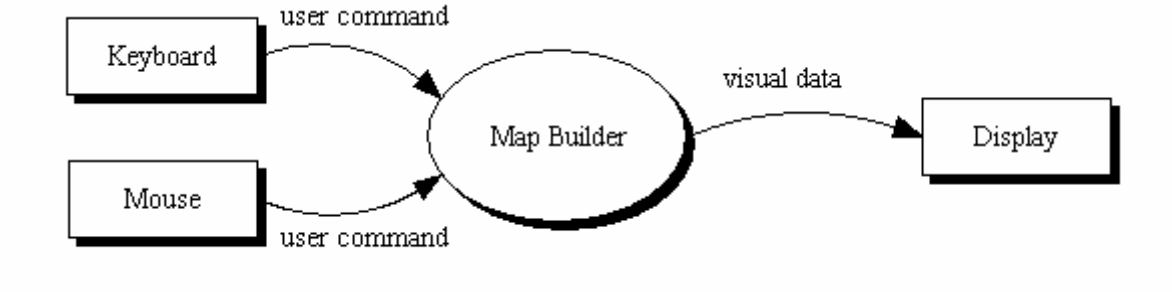


Figure 6.1 – DFD Level 0

The main program body, map builder, is decomposed into three modules in DFD Level 1 (See Figure 6.2). The three modules are namely, Graphical User Interface (GUI) Module, Data Module and Data Export Module.

GUI is responsible for interacting with the user to update the data structures in data module that hold the map environment data. Conversely, GUI gets the current map environment from the data module and displays in the user's monitor.

Secondly, the data module reads the state data of each model from the files previously created and stored in the same directory of the corresponding module. This module sends the map data to the data export module when the map file is to be saved.

Finally, the data export module is responsible for exporting the map data to a file with the extension ".map".

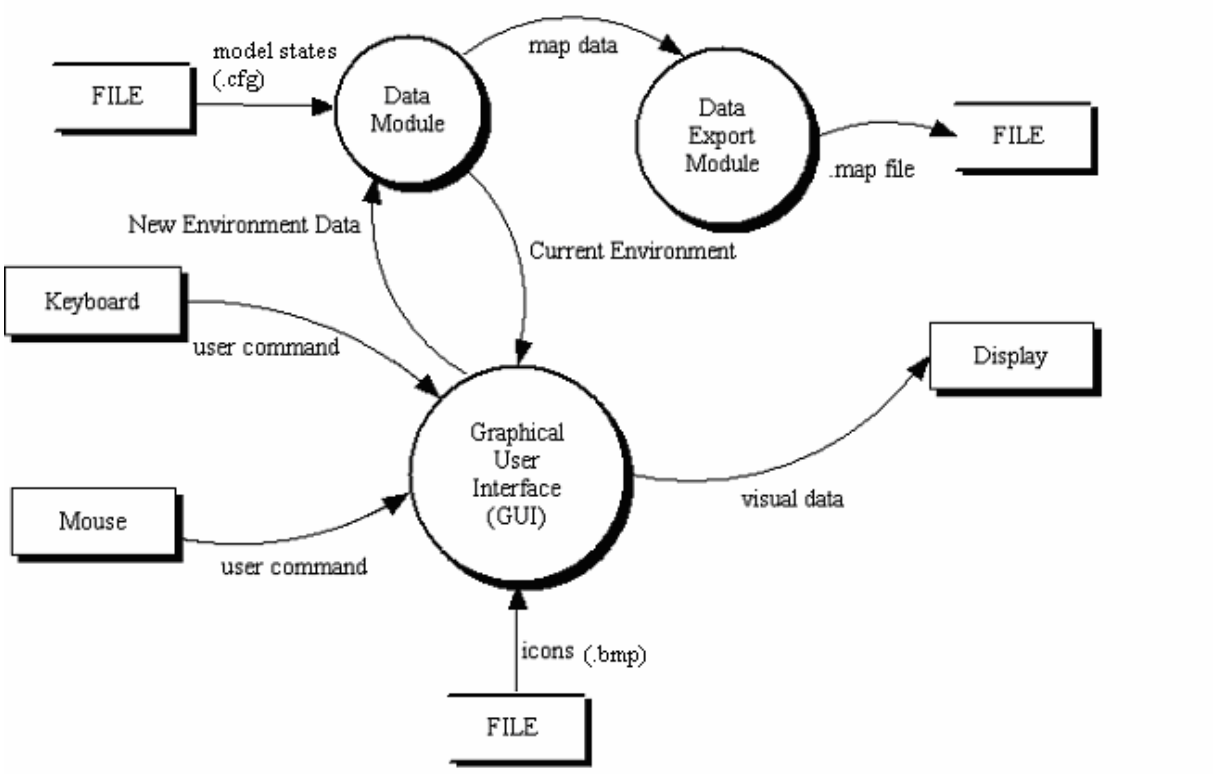


Figure 6.2 – DFD Level 1

The classes of the map builder software are depicted in Figure 6.3.

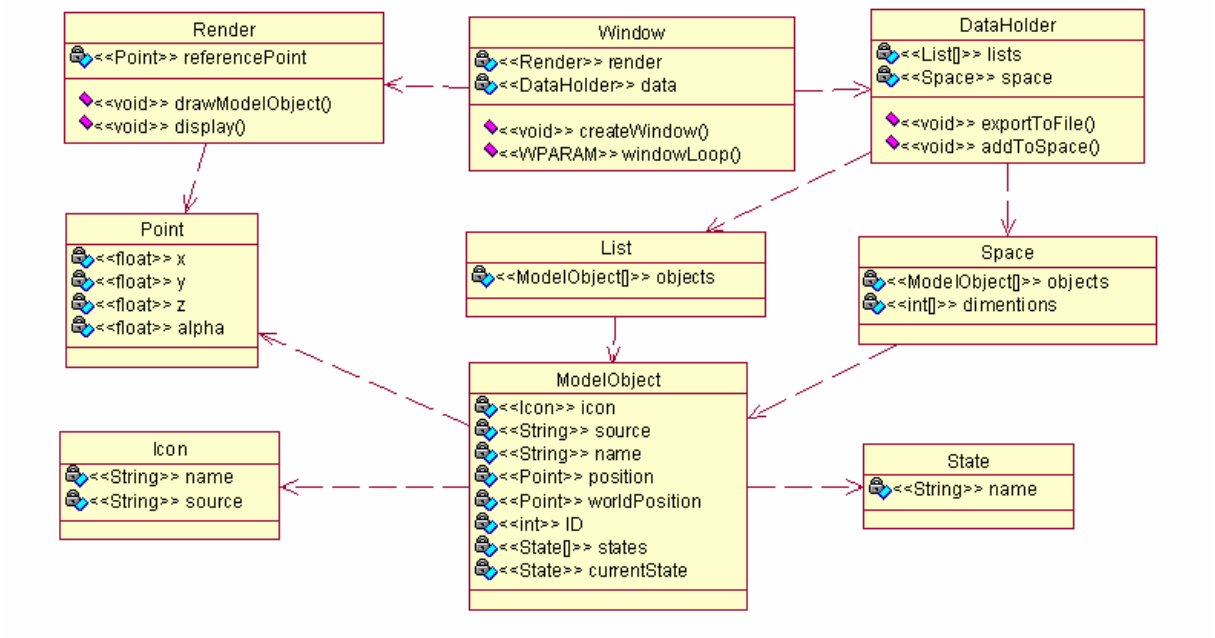


Figure 6.3 – Class Diagram for Map Builder

General view of the application will be as in the Figure 6.4. Through this window, the user will be able to locate the models into the map.

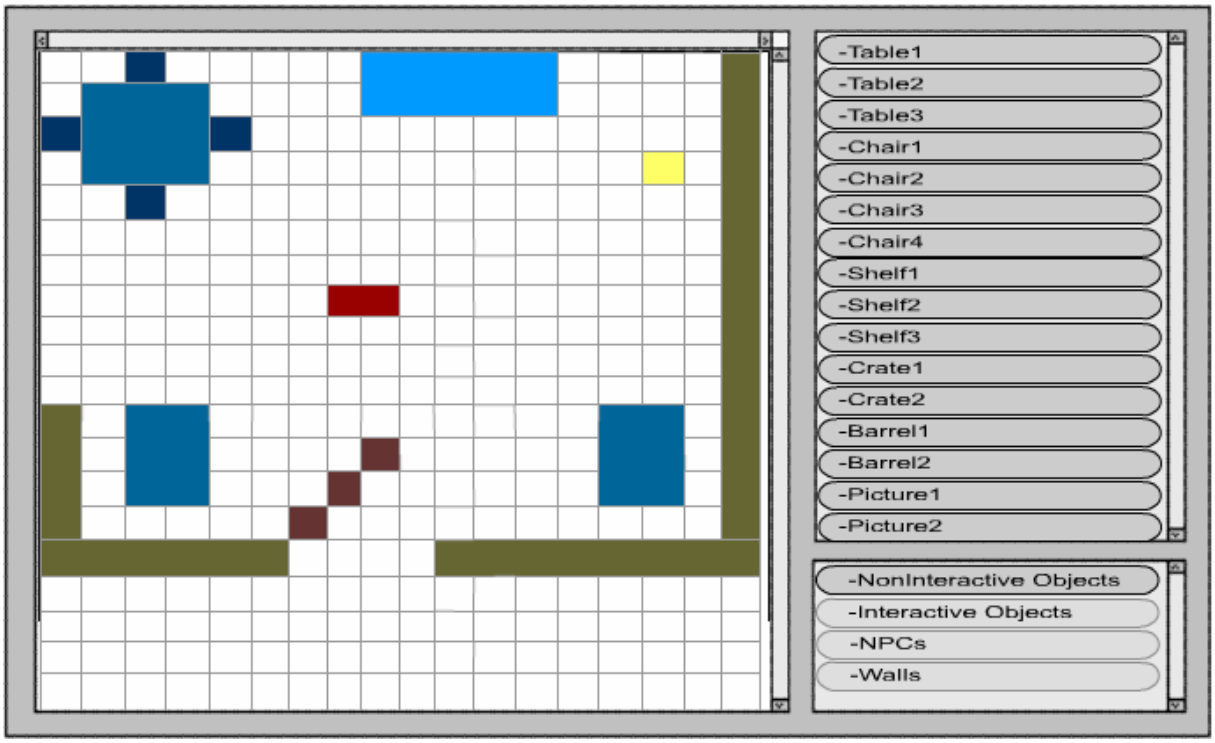


Figure 6.4 – Map Builder User Interface

7. GAME CONTENT

This section introduces the details of the game content to understand the general needs for the design of the system. The following are some of the key phrases identifying the main scenario:

- An evacuated uranium mine in Russia
- An old special force member is behind some unusual activity in the rafinery
- Suspected a deadly virus weapon in the mine
- A missing German agent
- A missing daughter of a Chinese businessman
- A British spy
- The mission of the British spy is to search for the missing German agent, and find out any unusual activity in the mine, learn who is behind it, and work to stop the activity.

The levels of the game are summarized in the following paragraphs. They are going to be the main consideration of the next section.

Start at the ground level of the mine; interrogate the mine technicians who are not aware of anything about the weapon and any other illegal activity. The hero has got a map of mine, but only for the first floor down. He will be alone most of the time because the mine has been evacuated already and something wrong is going on in the base. He needs to find a way to go down.

Stage starts with some trouble, at level 2, and at the beginning he witnesses some fire works. With a little search and observation, he comes to know that there happens some conflict down and he finds some bodies shot down. Searching the bodies turns him as information about the bad guys, and about some kidnap event. He learns that the kidnapped girl is a Chinese businessman's daughter, but when

he gets there she has already been killed. He also learns a little more about the virus, their intention about usage of the weapon. At this level, he comes across laser protected areas, magnetic pass doors, keys and alarm systems. Moreover, he needs to take down some guards and go through security doors. He needs to find a way to level 3.

At the final stage, the main task is to find the machine room of the rafinery and plant the bomb. At that level, the hero also finds out our missing agent's whereabouts. He finds his dead body. He pursuits the bad guy. Unfortunately, boss escapes, but still the hero neutralizes the weapon and plant the bomb, and he needs to get out fast. That last run will be the most thrilling one, through guns, he needs to make his way to the exit, he gets out and rafinery blows out. The mission completes.

As stated earlier, several missions, maps and so the puzzles will be able to be generated using map builder. Therefore, the mission under consideration here is a sample content to exemplify a possible game scenario. The next section concentrates on only the first level of the mission mentioned above. This mission will be the first one to be generated using the map builder.

7.1. Game Story

DETAILED DESCRIPTION OF LEVEL 1

The hero (X) gets a briefing from the agency about the operation on the phone. The code name of the operation is "Beyond the Sight".

X comes for some reported unusual activity in a uranium mine refinery in Russia. Agent X has two main objectives: first, finding out suspected terrorist activity and stop it; second, search for a missing German agent. Agency suspects about some deadly virus weapon and some old special force member behind the activity. X needs to infiltrate the base till sun light.

SCENE: Refinery Entrance – Midnight

Time is almost midnight and it is dark. X is watching the refinery from far. There looks weak security at the entrance, but still he must pass without being noticed, and therefore must be very careful. Gate is enlightened by spots. There are two armed people under one of these spots with no recognizable uniform. There is one more at the cabin near the entrance. There are crates and barrels scattered at the front, X thinks to make use of them. X also notices the electricity control panel at little distance from the entrance.

[X gets killed] X comes out openly, seen by the guards. Guards ask for id, and then fire. X falls dead.
[X makes guards leave their positions]
X puts out his pistol.

[If attempt to shot without silencer] – X thinks: "I better not to raise any attention".

He integrates the pistol with silencer. Aims at the electricity panel, which is at the left end of the screen. He takes his shot and following that power goes down. **X thinks: "Now, let's wait and see".**

One of the guards goes to look for the fuse box. X shots him down. X searches him, this turns in some ammo. After a minute other guard comes for looking. X shots him. **X thinks: "One more left".**

[If guard notices X] – Guard shots X dead.

He sneaks to entrance through shadows and strikes last guard down. Entrance is clear. He moves for the entrance gate.

SCENE: Entrance Lobby – later

X is at the lobby. There are two doors. X notices bulletin board, gets the floor map. **X thinks: "This is just what I need".**

Choice 1: North Door
X walks to the door in the north side of the room. There is a technician next to the door. X checks the door. The door is close. **X thinks: "I need the proper pass card".** X checks technician.

X:" I need to get into to the computer room. Can you get me there?"
Technician: "You must have level 1 pass card. There is nothing I can do".
X:"I come for NIA, National Inspection Agency, and need to see the computer center on the floor. I don't want any trouble. However, I got your collar id and I am planning to mention to your director about your extreme hospitality".
Technician:" Hey, I didn't know that. You may use my card. Here, take it".

X collects the pass card. X uses it on the door. Door is open.

Choice 2: West Door
X walks to the door in the west side of the room. X checks the door. Door is closed. **X thinks: "It is locked from the other side. It may be opened later".**

SCENE: North Door: Computer Room

X is at the computer room. There are computers and a server in the room. There is a door on the west wall. There is a large monitor at the north wall of the room. There is a technician working in front of the large monitor. X finds some cable on the floor. **X thinks:" That may come handy soon".** X finds a disc on the table. **X thinks: "That is a disc".** X checks the technician.

X:"I need to get to lift room".
Technician: "You better to see the security. Door is controlled from the Security Room. However, you can open it manually from there, in case you have the door entrance code".
X: "You may have some idea about this code?"
Technician: "The entrance code is changed in every two hours. That is all I know".

He moves to the door. X checks the door. A display panel appears. X needs to enter 4-digit entrance code.
[X gets caught] Entering invalid code three times raises the alarm. Security appears and catches X.
X leaves the Computer room and goes to Lobby.

SCENE: Entrance Lobby – later
X enters the lobby. The technician next to the door talks to X.

Technician: "Hello, again, now, Will you give my pass card back, please?

Choice 1
X:" [Lie] I lost it. It must be somewhere in the computer room"
Technician: "Oh, lost it? Thanks very much! (Technician leaves)

Choice 2
X: "Here is your card. I will talk about you to your director. What was your name, did you say? (X gives pass card)

Technician: "You don't need to do this. Hospitality is my nick name, eh. Umm, it is Johny Caster, if you ask.

The hero moves to the door on the west wall. X checks the door. Door is opened. X goes through.

SCENE: Personnel Dressing Room & Baths-later

X enters the room. There are cabinets for bath at the right half of the room. There are lockers at the left side. There is no one in the room. The lockers are enumerated from 201 to 220. X checks one of the lockers. It is locked.
X thinks: "It is locked." X finds a screwdriver on the floor.
X thinks: "That is a screwdriver."
[If X try to use screwdriver on the lockers] X thinks: "I don't want to do this." The hero moves for the door at the west wall of room. X checks the door. Door is opened. He moves in.

SCENE: Supply Stock Room- later

X enters the room. There is a technician at the far corner of the room. There are sacks of foods, and there is also a medical shelf on the left wall. There are cans of food at the left corner of room. X finds a can-opener on the floor.
X thinks: "That looks like a can-opener." X finds some pain-killer on the medical shelf.
X thinks: "I feel like I am going to need those, I am afraid."
X checks fat technician.

X: "What are you doing here?"
Technician: "Nothing. I was arranging these cans; yes that is what I am doing."
X: "You are just gobbling, as it looks from here. It is not lunch time, and that may be something to be reported."
Technician: "Ok, wait, you got me. Is there a way of compromise, huh?"
X: "Always. What can you offer, buddy?"
Technician: "Take these tokens. You may use them, these are for coke fridge. And forget what you saw, deal?"
X: "That's a deal, then."

X collects the tokens. He moves to door at the west wall. X checks the door. Door is opened. He moves in.

SCENE: Personnel Cafeteria- later

X enters the room. There are tables and chairs in the cafeteria. Cafeteria is dark at this time in the night. There is a switch left of the door. X checks the switch.
[If switch is not turned on] X doesn't move and check anything except switch. **X thinks:" I need light."**
It turns on. Room is enlightened. **X thinks:" That looks better."** There is a technician sitting on a chair. X checks technician.

X: "What is your problem, buddy?"
Technician: "Leave me alone, I have enough to worry about."
X: "May be I can help you, if you describe that problem a little, will you?
Technician: "Problem is all that rafinery, damn it. There is a rumor around that facility will be shut. I will lose my job."

X: "Are you sure about that?"
Technician: "Already some technicians have been sent on leave. "
X: "Is there someone I talk to about that? You should tell me, and I will see what I can do.
Technician: "Oh, thanks very much. One of my friends, George, has some information sources. I don't know about them. He can tell you what you want to hear. Say my name, which is Marcio. You can find him in the next room. "
The hero moves to the door leading to next room. X checks the door. Door is opened. He moves in.

SCENE: Personnel Rest Room

X enters the room. There is a billiard table in the middle of the room. There is a television at the northwest corner of the room. There are some technicians in the room. There are two technician watching TV, and two chatting next to billiard table. One of the technicians watching TV is George, the one Marcio mentioned. There is a packet of cigarettes on the table. X collects the packet.
X thinks: "I am trying to quit this thing, but that may come in handy later." X checks the technicians next to billiards.

Technician:" I am not in the mood to talk to you right now."

X checks the technician watching TV.

Technician: "I need cigarettes, will you lend me one?"
[If X doesn't have the packet conversation is over]
X: "Take the packet, I am working on to quit that thing."
Technician: "Sometimes I think of leaving it. I hope you make it. Thank you for cigarettes. Umm, I want to give you something for this. Take this pen, I have one another."

X collects the pen.

He moves to George and checks George.

George: "What do you want?"
[If X didn't talk to Marcio conversation is over]
X:"I am a friend of Marcio. He told me you may help me"
George: "Certainly you are told correct. What do you look for?"
X:"I heard some rumors. Facility may close its doors soon"
George: "That may be right. I don't know much thing about that issue. I hope it was just a trifling rumor.
X:"I need to get to Security room on the level. Can you say anything about that issue, then?"
George: "The door on east wall leads to that room."

The hero moves to door. He checks the door. Door is beeped. It is closed.
X thinks: "I need a proper pass card.".
[If X has level 1 pass card, and uses it] X thinks: "That doesn't work, I need the right key." He moves to George. He checks George.

George: "Now what?"
X:"I lost my pass card. I need to get to Security Room. Will you lend me yours?"
George: "No way I will do this."
The hero moves to door leading to personnel cafeteria. X checks door. Door is opened. He moves in.

SCENE: Personnel Cafeteria-later

X enters the room, moves to Marcio, and checks Marcio.

Marcio: "Yes?"
X:"I need to pass something to George, but I don't want to do this face-to-face. What is his lockers number?"
Marcio: "It is 206. "

He moves to door leading to stock room. X checks door .Door is opened. He moves in.

SCENE: Supply Stock Room-later

X enters the room.

[If X checks the storeman]
Storeman: "I am not in the mood to talk to you right now."

X moves the door leading to Dressing Room. X checks the door. Door is opened. He moves in.

SCENE: Personnel Dressing Room & Baths

X enters the room. He moves to locker 206. X uses screwdriver with the locker. It is opened. X collects a pass card. **[X loses screwdriver]**
X thinks: "That must be the pass card I need."
The hero moves to door leading to Stock room. He checks the door. It is opened. He moves in.

SCENE: Personnel Cafeteria

X enters the room. X moves the other door. X checks it. Door is opened. He moves in.

SCENE: Personnel Rest Room

He moves to other door, which leads to security room. X uses the pass card. The door is opened. He moves in.

SCENE: Level1 Security Room

X enters the room. There are computers as two lines at the north and south sides in the room. There is a key showcase on the west wall. X finds a key in the showcase. **X thinks: "That is a key, with a number on it, 207 or 209, I can't read it."**
[X gets caught] If X moves into middle of the room, security will see X. They ask for id. X gets caught.
He moves to door back. X checks the door. Door is opened. He moves in.

SCENE: Rest Room

X passes to cafeteria.

SCENE: Cafeteria

X passes to stock room.

SCENE: Stock Room

X passes to dressing room.

SCENE: Dressing Room & Baths

X enters the room.
[X uses the key on locker 207] X thinks: "That is not going to get opened."
[X uses the key on locker 209] Locker is opened. X collects a password decoder. **X thinks: "That gives me an idea."**
He moves to door leading to Lobby. He checks the door. It is opened. He moves in.

SCENE: Entrance Lobby

He moves to door on the north wall.

[If X has the level1 pass card] The technician has gone. X uses pass card. Door is opened. He moves in.
[If X does not have] X checks the technician.

Technician: "Nice to see you, sir"
X:"I need to get to next room. Can you help?"
Technician: "I can, sir. But first, I need a pen or something to sign my papers. Did you happen to have any?"
[If X does not have the pen, conversation is over]
X: "That may work."
Technician: "Thanks very much. You should use my card mister."

X loses the pen. X collects the level1 pass card. X uses it on the door. Door is opened. He moves in.

SCENE: Computer Room

He moves to the door leading to lifts. X uses decoder on the door. Door is opened. He moves in.

Level Completed

General Things

[If something irrelevant is used on somewhere in the game (like using disc on a locker)] X thinks:

"That won't work."
"Irrelevant."
"That is not logical."
"No way."
"Try to think something else."
"...umm..no…"

[If X checks some NPC whose duty in the game is over (like the one that was in the Stock room)] Technician:

"I am not in the mood to talk to you right now."
"I am too busy."
"I don't feel like chatting, sorry."
"…."

7.2. Motions and Actions of Hero

The followings are the movements and actions that the main character is capable of:

- Walk (By direction buttons/ by right mouse click in "walk mode")
- Run (By direction buttons/ by right mouse click in "run mode")
- Crouch (By direction buttons/ by right mouse click in "crouch mode")
- Use (an item) (By left mouse click)
- Pick (By left mouse click in "stealth mode")
- Talk (By left mouse click in "stealth mode")
- Aim (By left mouse click in "fire mode")
- Die (There are several ways ;)
- Idle Action ("Scratching back" after being idle for sometime)

The transitions between these motion states are schematized in Figure 7.1.

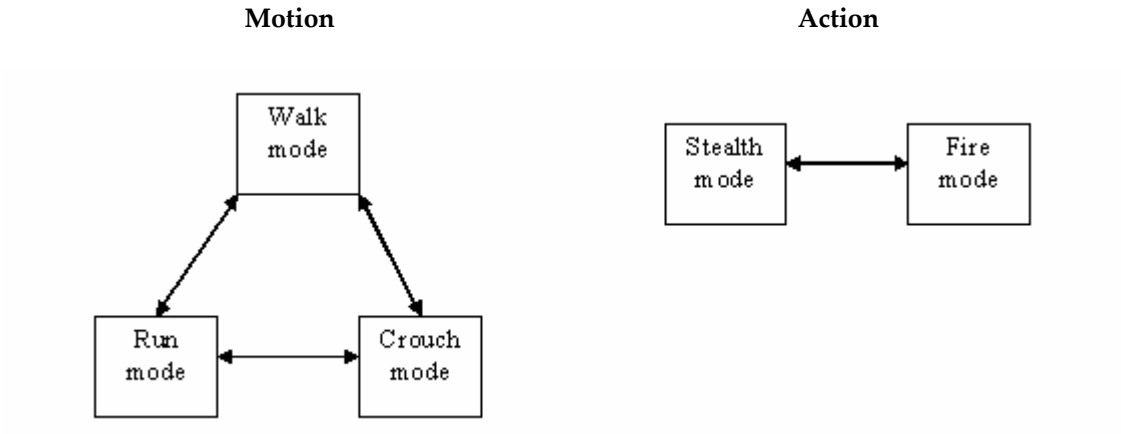


Figure 7.1 – Hero's motion and action states

7.3. Detailed In-Game State Transitions

Here the entities: objects, characters and equipments in the game flow are discussed. The game flow is schematized in the two Figures 7.2 and 7.3 in terms of all the entity interactions. The entities are indexed as follows to ease the understanding.

NPCs
Tech_1: Technician in the Lobby (Johny)
Tech_2: Technician in the Computer Room
Tech_3: Technician in the Stock Room
Tech_4: Technician in the Cafeteria (Marcio)
Tech_5: Technician in the Rest room, next to billiards
Tech_6: Technician in the Rest room, next to billiards
Tech_7: Technician in the Rest room, smoker
Tech_8: Technician in the Rest room (George)
Grd_1: Guard outside
Grd_2: Guard outside
Grd_3: Guard outside, at the cabin

Doors
Door_1: North door, Lobby
Door_12 South door, Computer room
Door_2: West door, Lobby
Door_22 East door, Dressing room
Door_3: West door, Computer room
Door_4: West door, Dressing room
Door_42 East door, Stock room
Door_5: West door, Stock room
Door_52 East door, Cafeteria
Door_6: North door, Cafeteria
Door_62 South door, Rest room
Door_7: East door, Rest room
Door_72 West door, Security room
Door_8: East door, Security room

Equipments-Items
Eq_1: pistol
Eq_2: silencer
Eq_21: silenced pistol
Eq_3: level1 pass card
Eq_4: disc
Eq_5: cable
Eq_6: can-opener
Eq_7: screwdriver
Eq_8: tokens
Eq_9: cigarettes
Eq_10: locker key #209
Eq_11: decoder
Eq_12: map
Eq_13: pain-killer
Eq_14: pen
Eq_15: security pass card

Other Items
O_1: Electricity box
O_2: Locker #206
O_3: Locker #209
O_4: Locker XXX
O_5: Switch
O_6: Key showcase

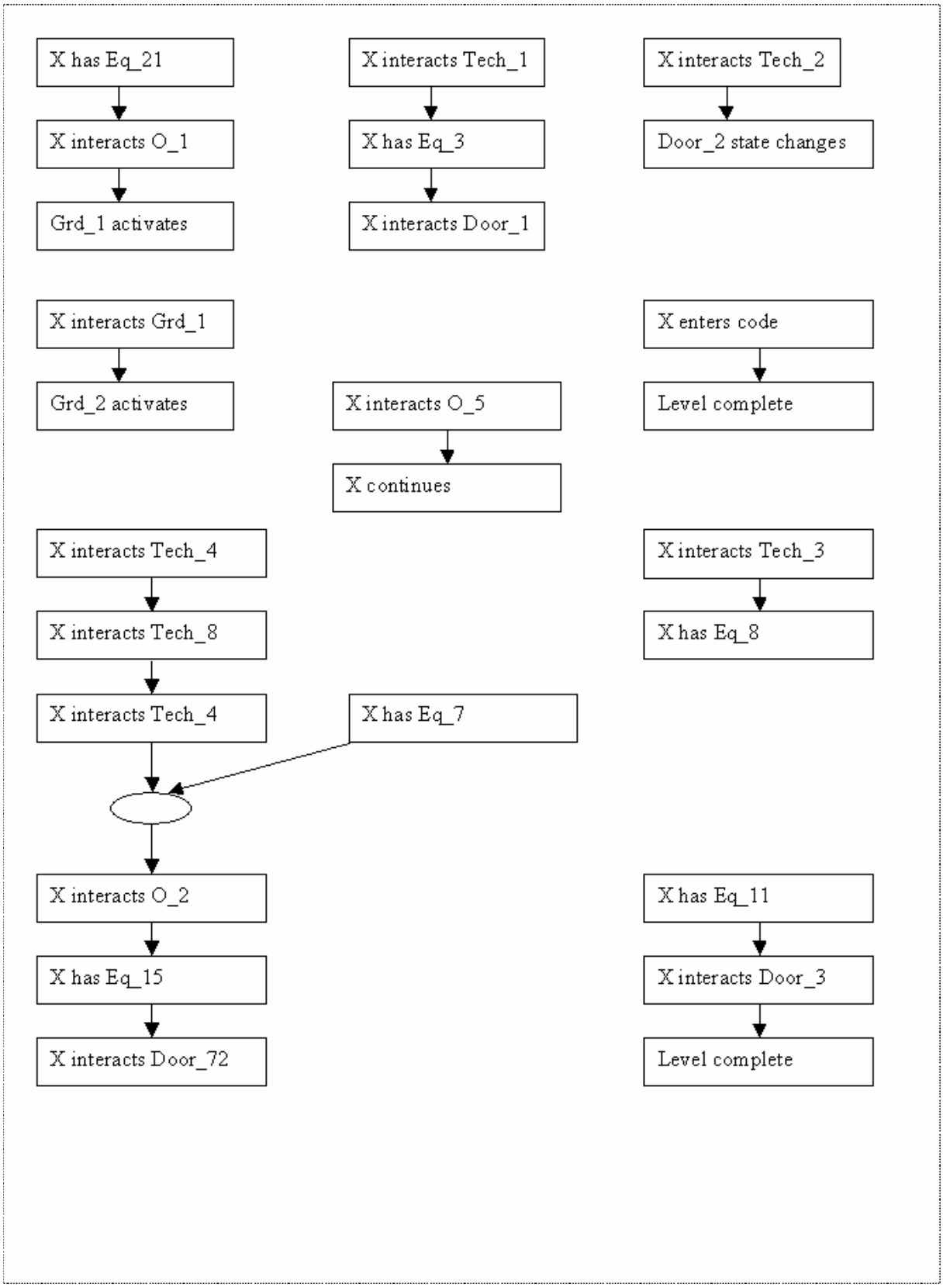

X has Eq_21
X interacts O_1
Grd_1 activates
X interacts Tech_1
X has Eq_3
X interacts Door_1
X interacts Tech_2
Door_2 state changes
X interacts Grd_1
Grd_2 activates
X interacts O_5
X continues
X enters code
Level complete
X interacts Tech_4
X interacts Tech_8
X interacts Tech_4
X has Eq_7
X interacts Tech_3
X has Eq_8
X interacts O_2
X has Eq_15
X interacts Door_72
X has Eq_11
X interacts Door_3
Level complete

Figure 7.2

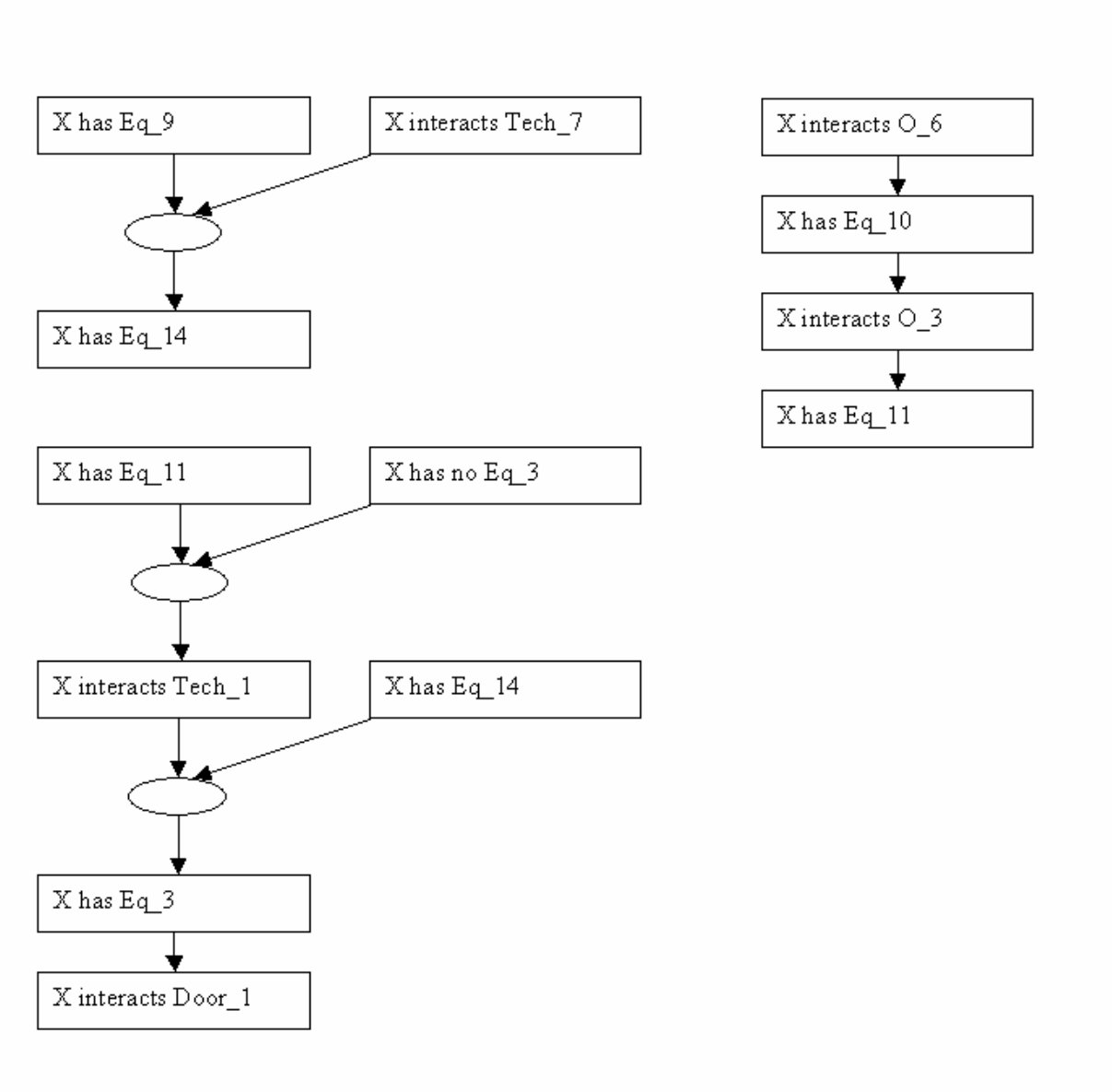


Figure 7.3

7.4. User Interface

Below is presented the inventory interface that will cover some area at the bottom part of the game screen (See Figure 7.4). The game states depend on the activation of objects in that interface; user will keep its items here and see them while playing the game. There are two types of slots for collected items. When a new item is collected, Normal slots are used. In order to combine two items if allowed, user will pass two items from the normal slots to combination slots. In order to pass an item between slots, first an item to be passed should be activated. Below you will see an active slot, which is ready for a possible pass. More behaviors and interdependencies between game and inventory screen are completely stated in the state diagrams section.

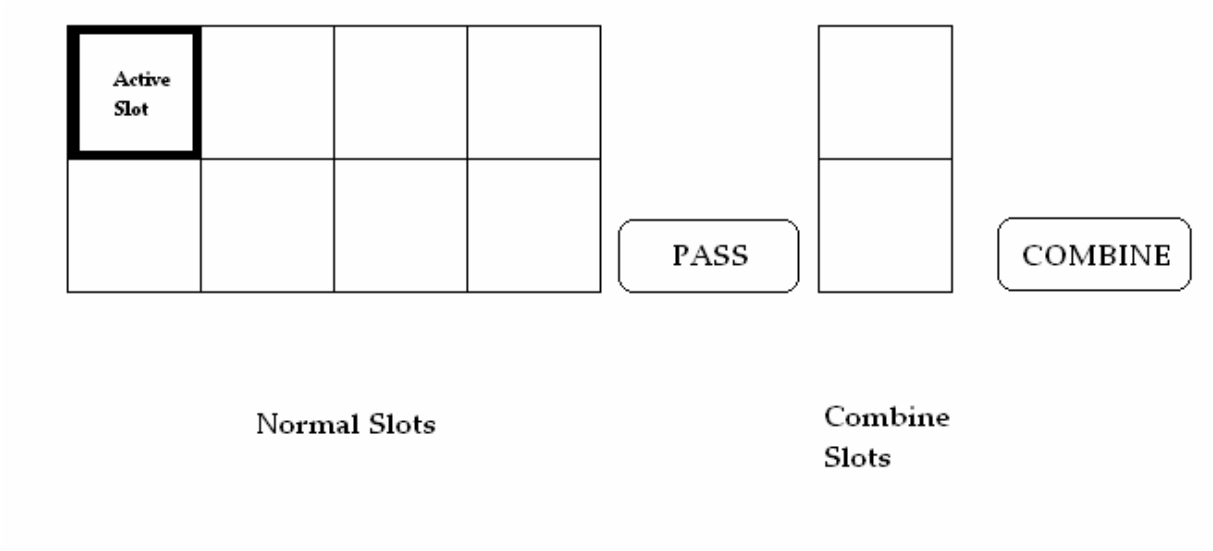


Figure 7.4 – Inventory Screen

8. CONCLUSION

This report gives detailed design specifications of the game software 'Beyond The Sight' and is a baseline for the final design specification and game prototype implementation. The subjects whose design details are not mention properly enough in this report will be concentrated in the final design report.

The time schedule for the development progress is given in the Appendix Part. The tasks are assigned to the following roles:

Management team:	charged with general aspects of the system definition and concepts.
Reporter:	charged with preparation of project reports.
Researcher:	charged with research activities on the technical issues.
Project Manager:	charged with making critical decisions.
Analyst:	charged with analyzing the requirements.
Developer:	charged with designing and developing the software.
Art Director:	charged with artistic work of the game like scenario.
Revisor:	charged with tuning the designed components.
Presentation Team:	charged with preparation of presentations.
Tester:	charged with testing the software.
Technical Communicator:	charged with developing the software documentation.

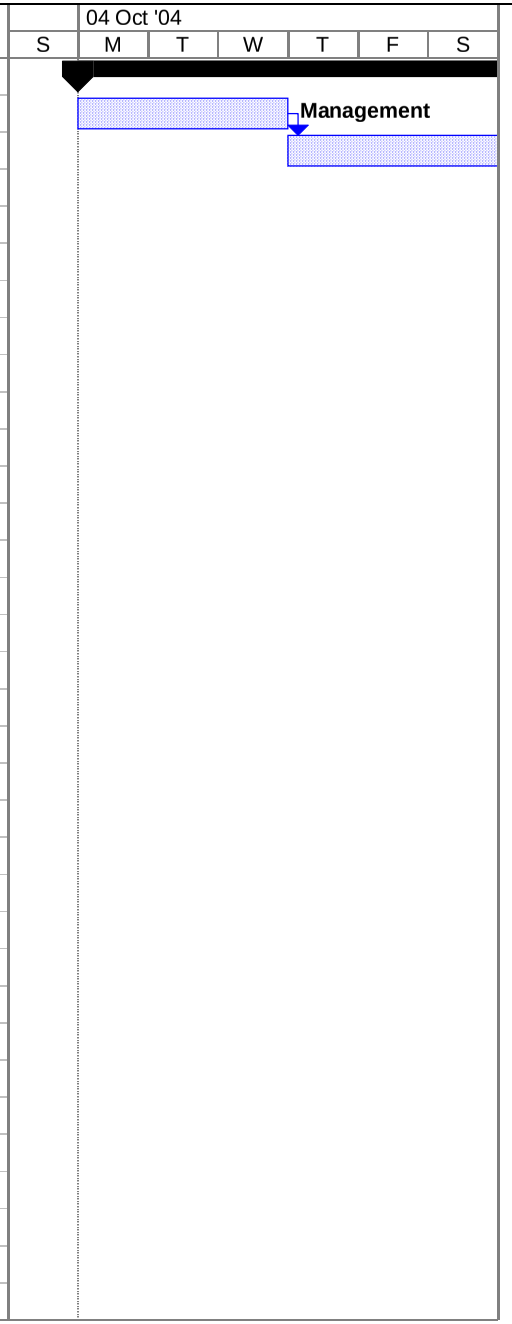
The roles and responsibilities are subject to change among the team members to provide the members with the opportunity to experience all kind of duties through the project development. When necessary, external support will be concerned (i.e. Testers).

9. REFERENCES

1. Pressman, R.S., *Software Engineering* – A Practitioner's Approach, 5th Edition, p. 28
2. Pressman, R.S., *Software Engineering – A Practitioner's Approach,* 5th Edition, p. 311
3. Pressman, R.S., *Software Engineering – A Practitioner's Approach,* 5th Edition, p. 317

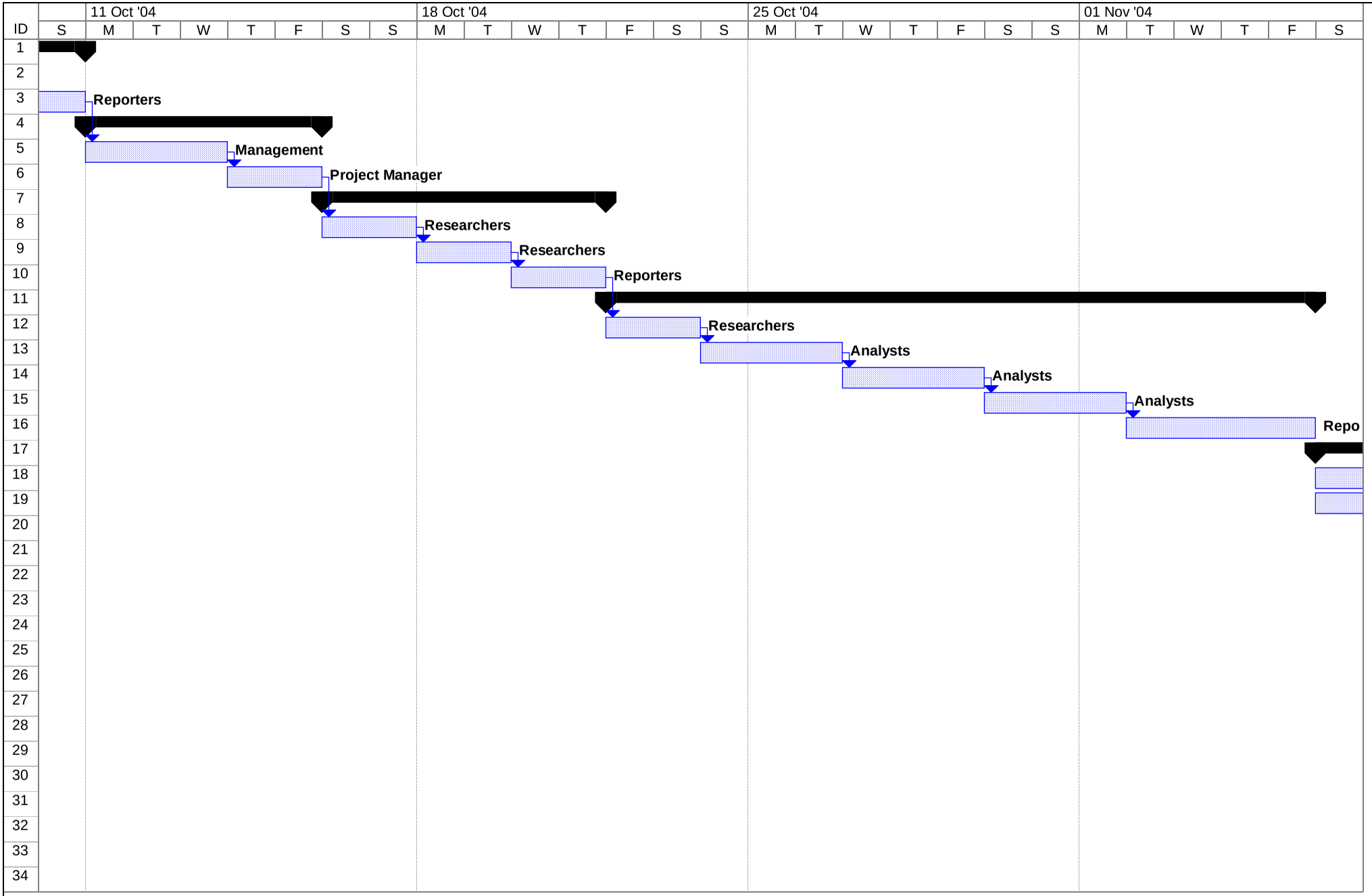
IEEE Recommended Practice for Software Design Descriptions, 23 September 1998

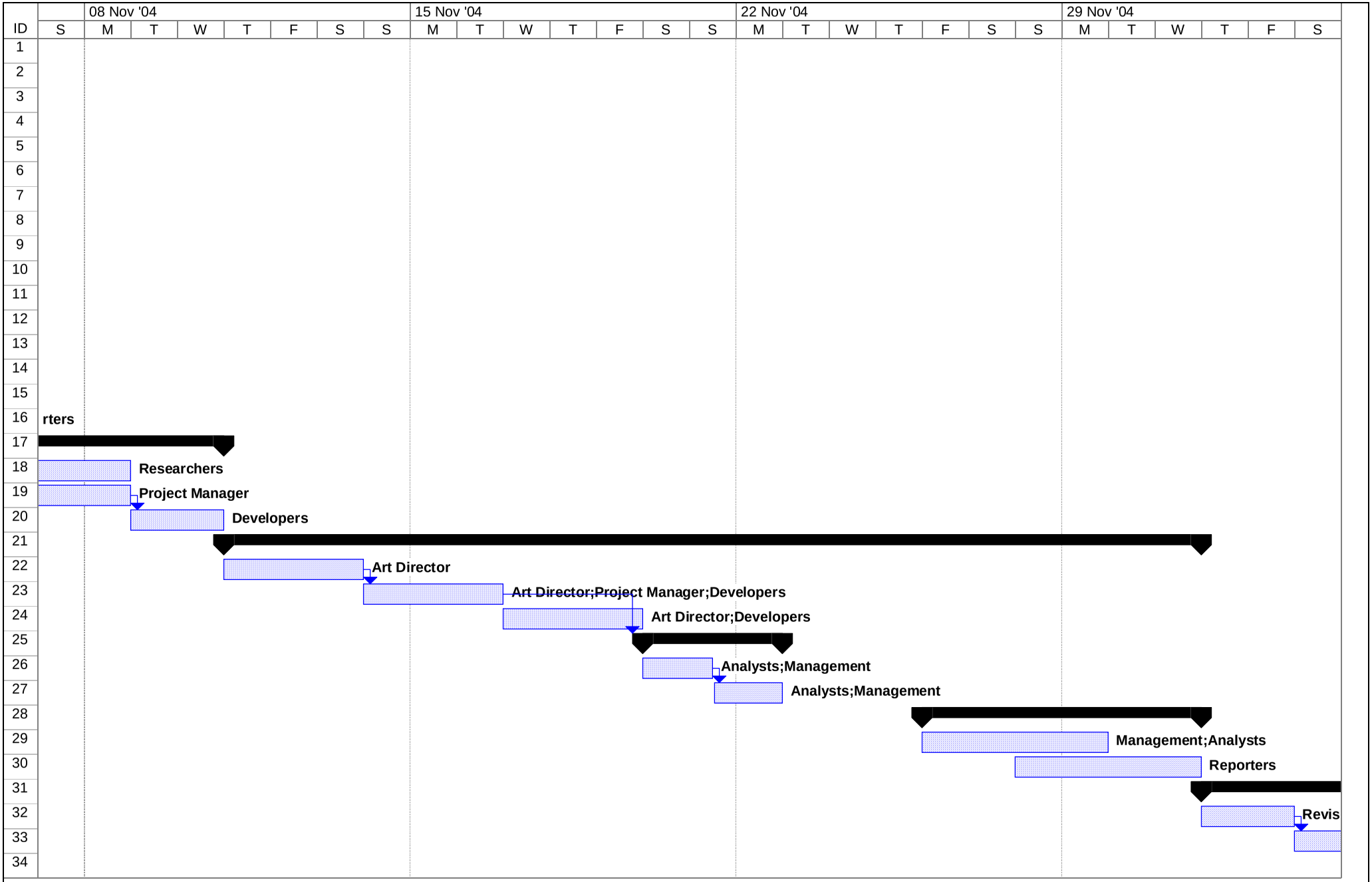
ID	Task Name	Duration	Start	Finish	Resource Names
1	**Team Construction**	**7 days**	**Mon 04.10.04**	**Sun 10.10.04**	
2	Problem Definition	3 days	Mon 04.10.04	Wed 06.10.04	Management
3	Proposal Report Preparation	4 days	Thu 07.10.04	Sun 10.10.04	Reporters
4	**Scope**	**5 days**	**Mon 11.10.04**	**Fri 15.10.04**	
5	Determination of Project Scope	3 days	Mon 11.10.04	Wed 13.10.04	Management
6	Conclusion on The Project Scope	2 days	Thu 14.10.04	Fri 15.10.04	Project Manager
7	**Research and Survey**	**6 days**	**Sat 16.10.04**	**Thu 21.10.04**	
8	Search on General Features	2 days	Sat 16.10.04	Sun 17.10.04	Researchers
9	Literature Survey (Game Comparisons)	2 days	Mon 18.10.04	Tue 19.10.04	Researchers
10	Survey Report Preparation	2 days	Wed 20.10.04	Thu 21.10.04	Reporters
11	**Requirement Analysis**	**15 days**	**Fri 22.10.04**	**Fri 05.11.04**	
12	Gathering Requirements	2 days	Fri 22.10.04	Sat 23.10.04	Researchers
13	Req. Specifications Draft	3 days	Sun 24.10.04	Tue 26.10.04	Analysts
14	Conclusion on Req. Specifications	3 days	Wed 27.10.04	Fri 29.10.04	Analysts
15	Revision of Requirements	3 days	Sat 30.10.04	Mon 01.11.04	Analysts
16	Requirement Analysis Report Preparation	4 days	Tue 02.11.04	Fri 05.11.04	Reporters
17	**Development Tools**	**5 days**	**Sat 06.11.04**	**Wed 10.11.04**	
18	Research on Market Tools	3 days	Sat 06.11.04	Mon 08.11.04	Researchers
19	Risk Estimations	3 days	Sat 06.11.04	Mon 08.11.04	Project Manager
20	Conclusion on Development Tools	2 days	Tue 09.11.04	Wed 10.11.04	Developers
21	**Game Design**	**21 days**	**Thu 11.11.04**	**Wed 01.12.04**	
22	Scenario	3 days	Thu 11.11.04	Sat 13.11.04	Art Director
23	Game Play Design	3 days	Sun 14.11.04	Tue 16.11.04	Art Director;Project Manager;Developers
24	User Interface Design	3 days	Wed 17.11.04	Fri 19.11.04	Art Director;Developers
25	**Multimedia Design**	**3 days**	**Sat 20.11.04**	**Mon 22.11.04**	
26	Graphics Concepts	1,5 days	Sat 20.11.04	Sun 21.11.04	Analysts;Management
27	Audio Concept	1,5 days	Sun 21.11.04	Mon 22.11.04	Analysts;Management
28	**Initial Level Design**	**6 days**	**Fri 26.11.04**	**Wed 01.12.04**	
29	Level Design Work	4 days	Fri 26.11.04	Mon 29.11.04	Management;Analysts
30	Initial Design Report	4 days	Sun 28.11.04	Wed 01.12.04	Reporters
31	**Game Design Tuning**	**12 days**	**Thu 02.12.04**	**Mon 13.12.04**	
32	Scenario Tuning	2 days	Thu 02.12.04	Fri 03.12.04	Revisors;Developers
33	User Interface Design Tuning	3 days	Sat 04.12.04	Mon 06.12.04	Revisors;Developers
34	Level Design Tuning	5 days	Thu 09.12.04	Mon 13.12.04	Revisors;Developers

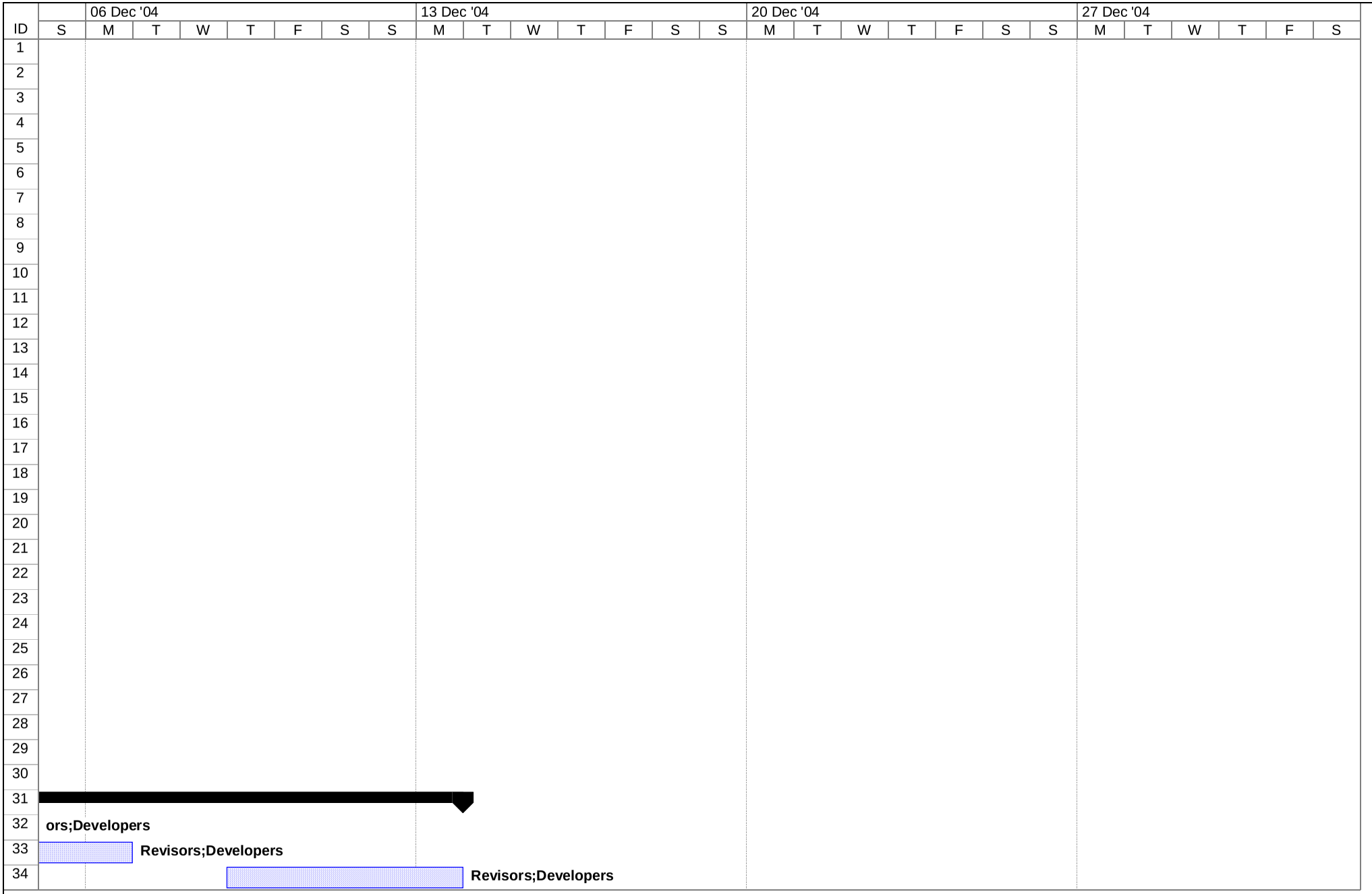


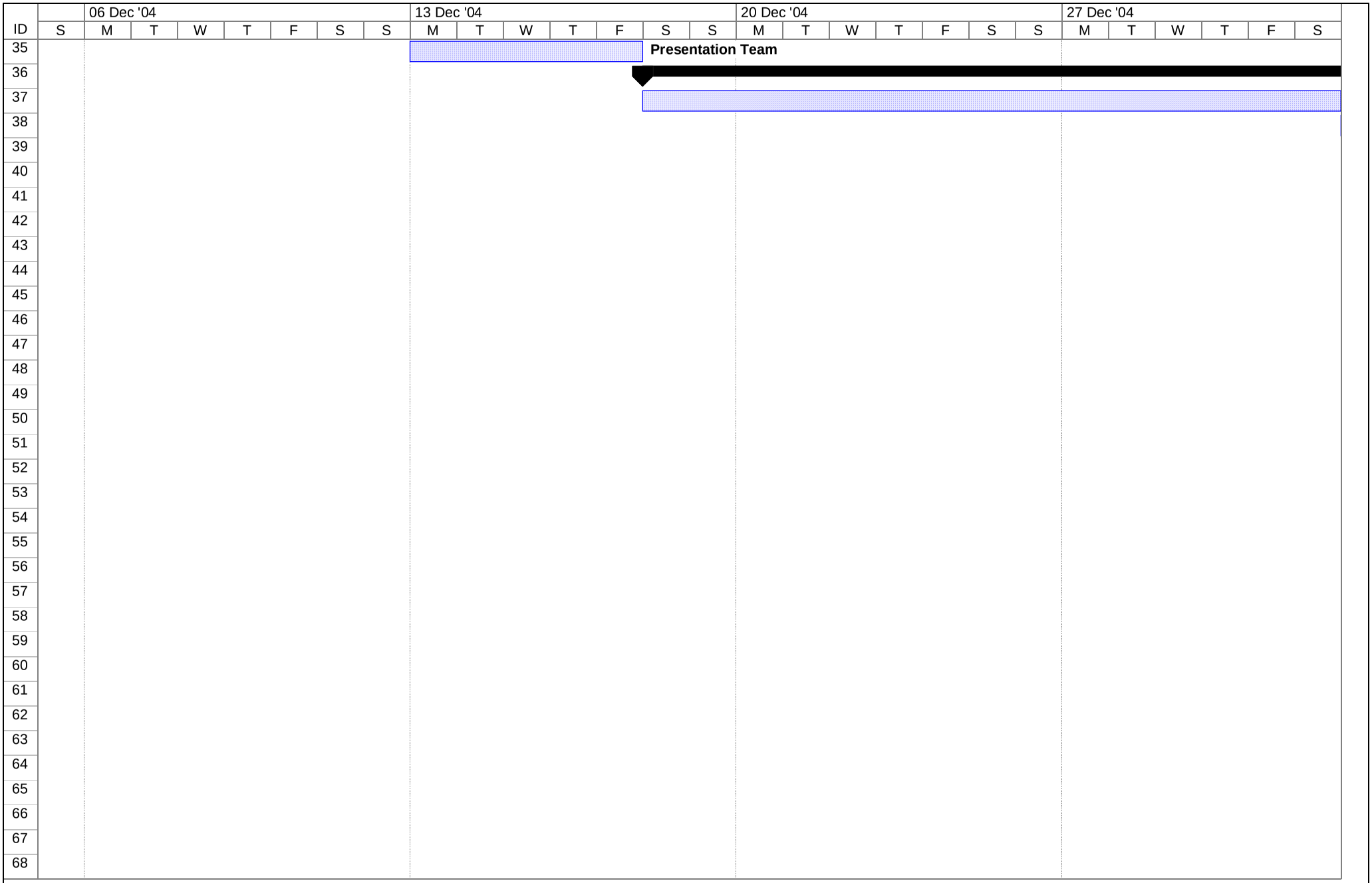
ID	Task Name	Duration	Start	Finish	Resource Names	04 Oct '04						
						S	M	T	W	T	F	S
35	Project Presentation	5 days	Mon 13.12.04	Fri 17.12.04	Presentation Team							
36	**Final (Detailed) Level Design**	**23 days**	**Sat 18.12.04**	**Sun 09.01.05**								
37	Final Design Work	15 days	Sat 18.12.04	Sat 01.01.05	Developers							
38	Final Design Report	8 days	Sun 02.01.05	Sun 09.01.05	Reporters							
39	Game Design Complete	1 day	Mon 10.01.05	Mon 10.01.05	Developers;Project Manager							
40	**Prototype**	**11 days**	**Sun 09.01.05**	**Wed 19.01.05**								
41	Prototype Preparation	10 days	Sun 09.01.05	Tue 18.01.05	Developers							
42	Demo	1 day	Wed 19.01.05	Wed 19.01.05	Presentation Team							
43	**Multimedia Implementation**	**10 days**	**Wed 16.02.05**	**Fri 25.02.05**								
44	Create Graphics	10 days	Wed 16.02.05	Fri 25.02.05	Developers							
45	Cretate Audio	10 days	Wed 16.02.05	Fri 25.02.05	Developers							
46	**Implementation**	**47 days**	**Sat 26.02.05**	**Wed 13.04.05**								
47	Basic Classes for Models	6 days	Sat 26.02.05	Thu 03.03.05	Developers							
48	Game Engine Implementation	6 days	Thu 03.03.05	Tue 08.03.05	Developers							
49	Graphics Engine and Sound Classes Imp.	6 days	Tue 08.03.05	Sun 13.03.05	Developers							
50	First Development Snapshot	1 day	Mon 14.03.05	Mon 14.03.05	Developers							
51	Menu and Configuration Classes Impl.	5 days	Tue 15.03.05	Sat 19.03.05	Developers							
52	AI Engine Implementation	5 days	Sun 20.03.05	Thu 24.03.05	Developers							
53	Integration	20 days	Fri 25.03.05	Wed 13.04.05	Developers							
54	**Map Builder Implementation**	**10 days**	**Thu 14.04.05**	**Sat 23.04.05**								
55	GUI Implementation	5 days	Thu 14.04.05	Mon 18.04.05	Developers							
56	Data Abstraction and Export Layers Impl.	5 days	Tue 19.04.05	Sat 23.04.05	Developers							
57	Integration of all Codes and Tuning	9 days	Sun 24.04.05	Mon 02.05.05	Developers							
58	First Release	1 day	Mon 02.05.05	Mon 02.05.05	Developers							
59	**Testing**	**17 days**	**Tue 03.05.05**	**Thu 19.05.05**								
60	Game Engine Testing	5 days	Tue 03.05.05	Sat 07.05.05	Testers							
61	**Initial Game Testing**	**12 days**	**Sun 08.05.05**	**Thu 19.05.05**								
62	User Interface Testing	4 days	Sun 08.05.05	Wed 11.05.05	Testers							
63	Audio Testing	2 days	Thu 12.05.05	Fri 13.05.05	Testers							
64	Game Play Testing	6 days	Sat 14.05.05	Thu 19.05.05	Testers							
65	Project Presentation	2 days	Fri 20.05.05	Sat 21.05.05	Presentation Team							
66	Final Testing	7 days	Sun 22.05.05	Sat 28.05.05	Testers							
67	Final Release	8 days	Sun 29.05.05	Sun 05.06.05	Developers							
68	**Documentation**	**4 days**	**Mon 06.06.05**	**Thu 09.06.05**								

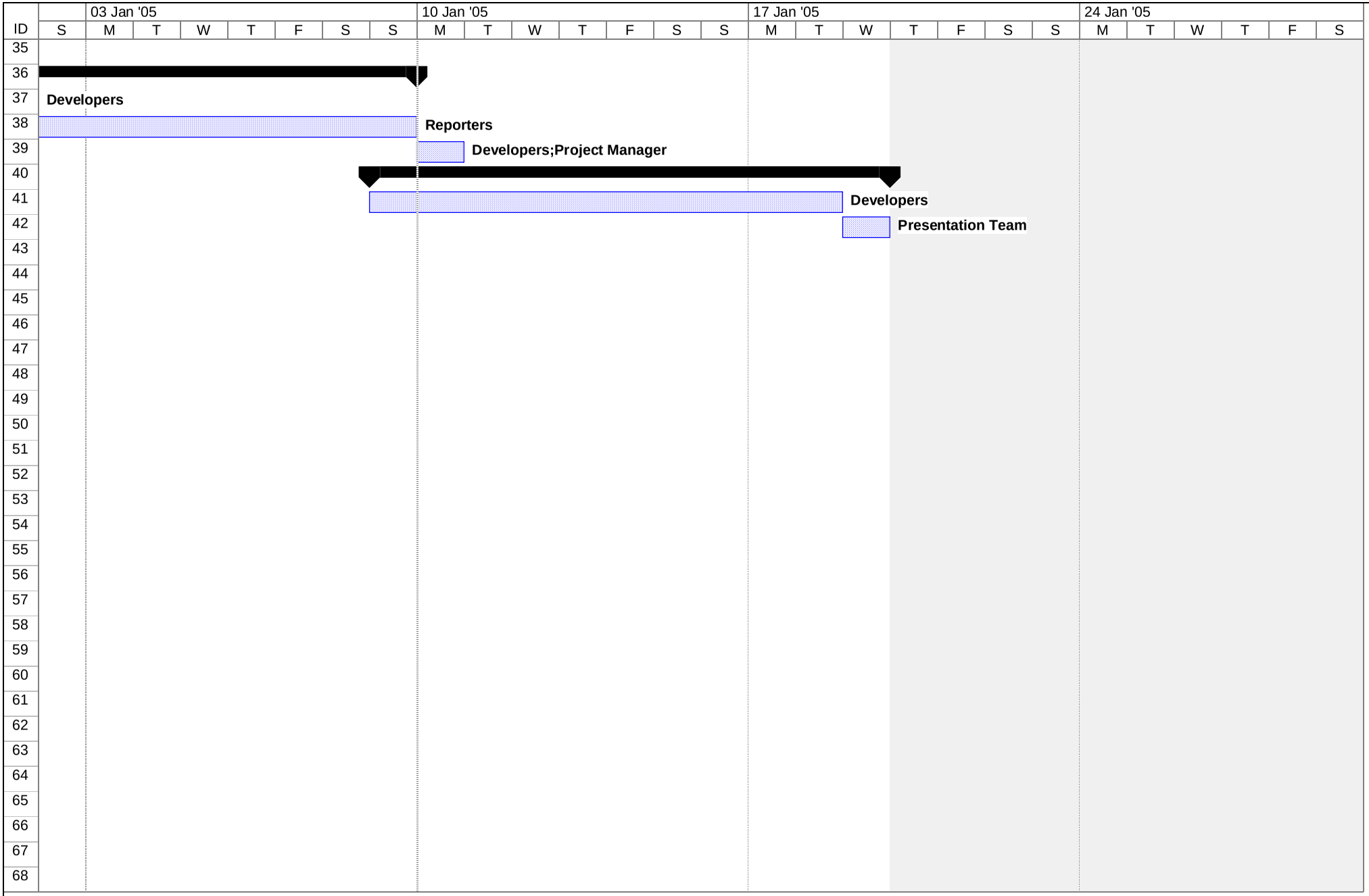
ID	Task Name	Duration	Start	Finish	Resource Names	04 Oct '04						
						S	M	T	W	T	F	S
69	User Manual	4 days	Mon 06.06.05	Thu 09.06.05	Technical Communicators							
70	Programmers' Documentation	4 days	Mon 06.06.05	Thu 09.06.05	Technical Communicators							
71	Final Demonstration	1 day	Fri 10.06.05	Fri 10.06.05	Presentation Team							

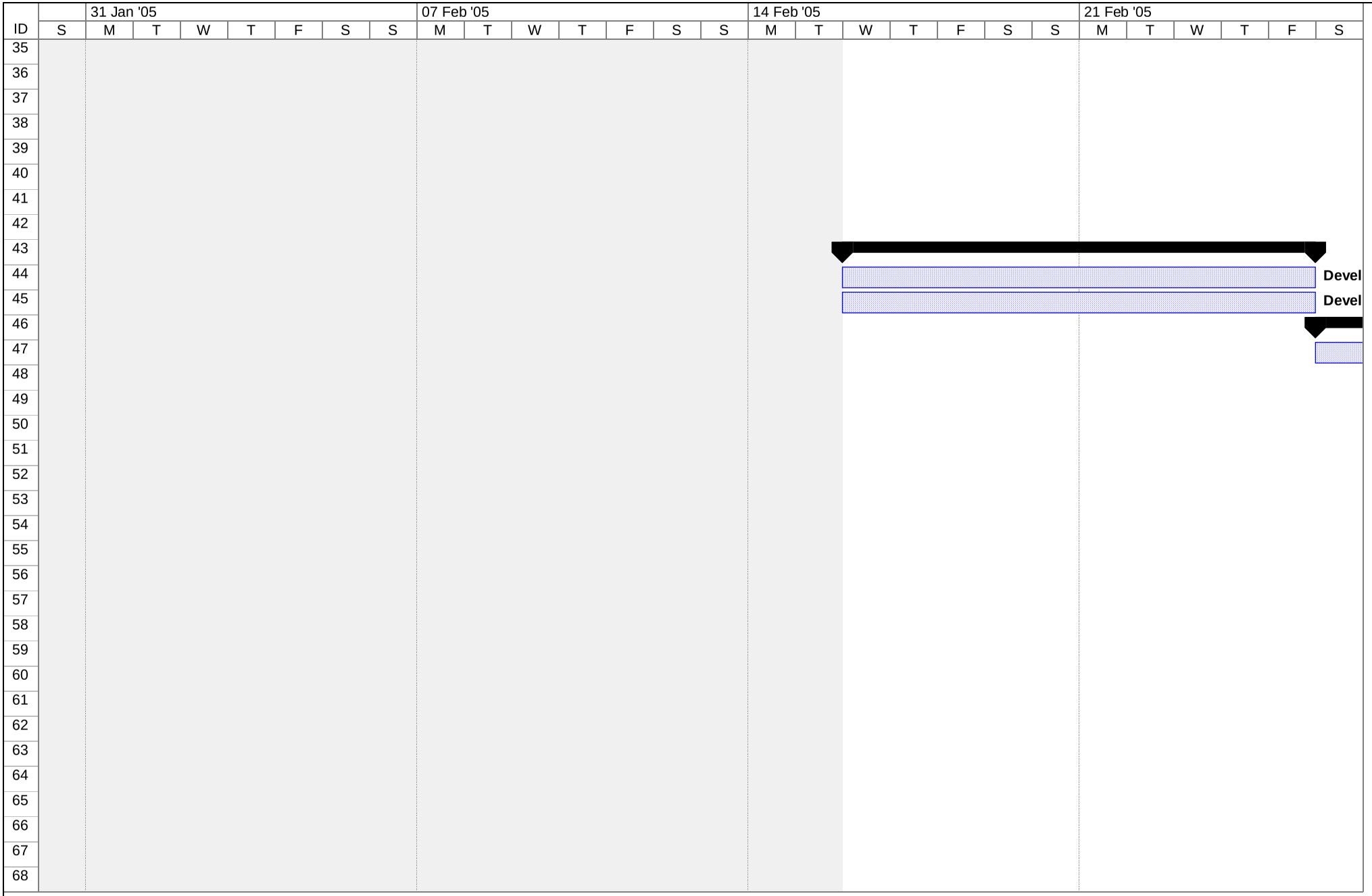

ID
11 Oct '04
18 Oct '04
25 Oct '04
01 Nov '04
S M T W T F S S M T W T F S S M T W T F S S M T W T F S
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
Reporters
Management
Project Manager
Researchers
Researchers
Reporters
Researchers
Analysts
Analysts
Analysts
Repo

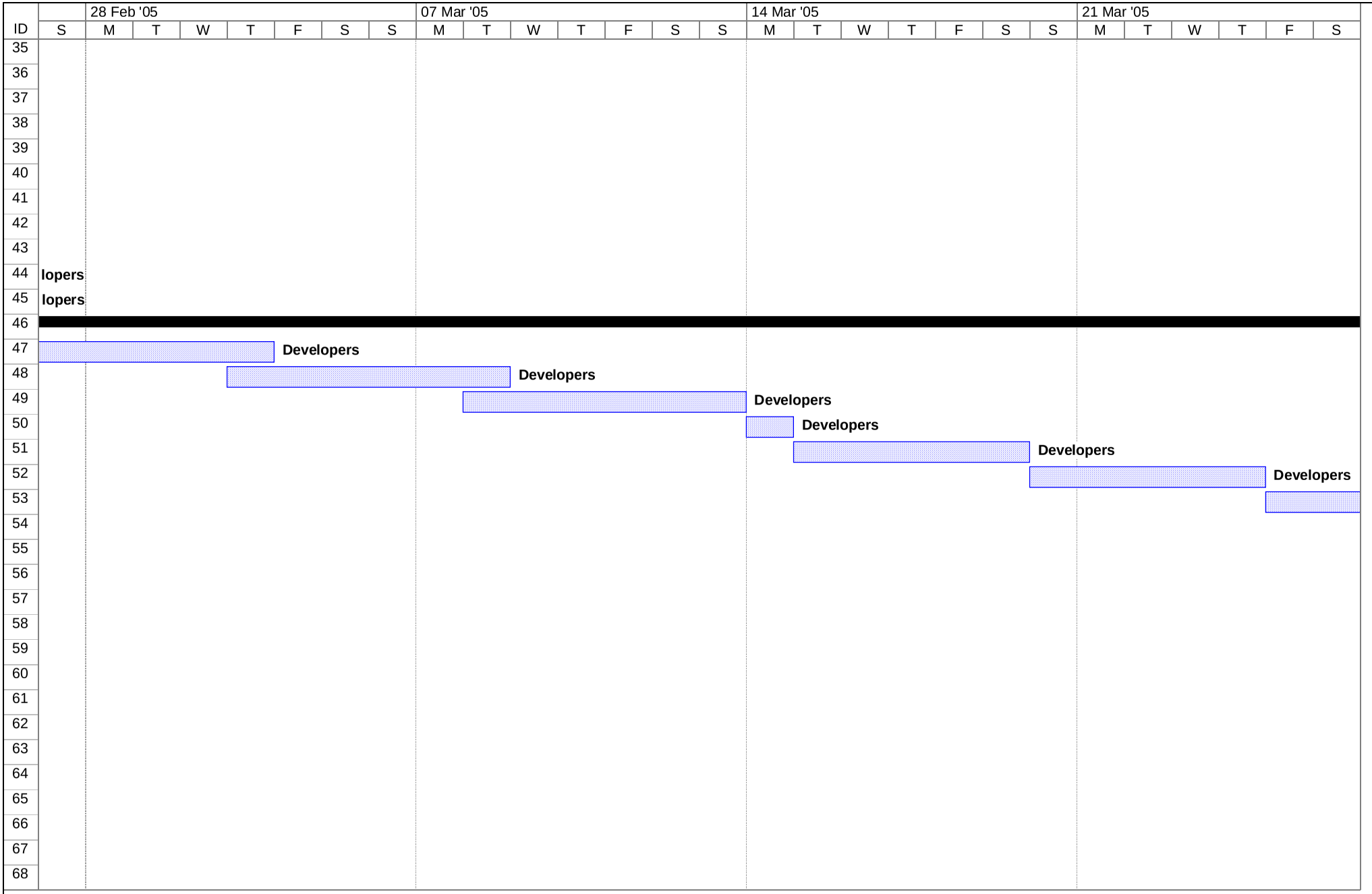

ID
08 Nov '04
15 Nov '04
22 Nov '04
29 Nov '04
S M T W T F S S M T W T F S S M T W T F S S M T W T F S
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
rters
Researchers
Project Manager
Developers
Art Director
Art Director;Project Manager;Developers
Art Director;Developers
Analysts;Management
Analysts;Management
Management;Analysts
Reporters
Revis

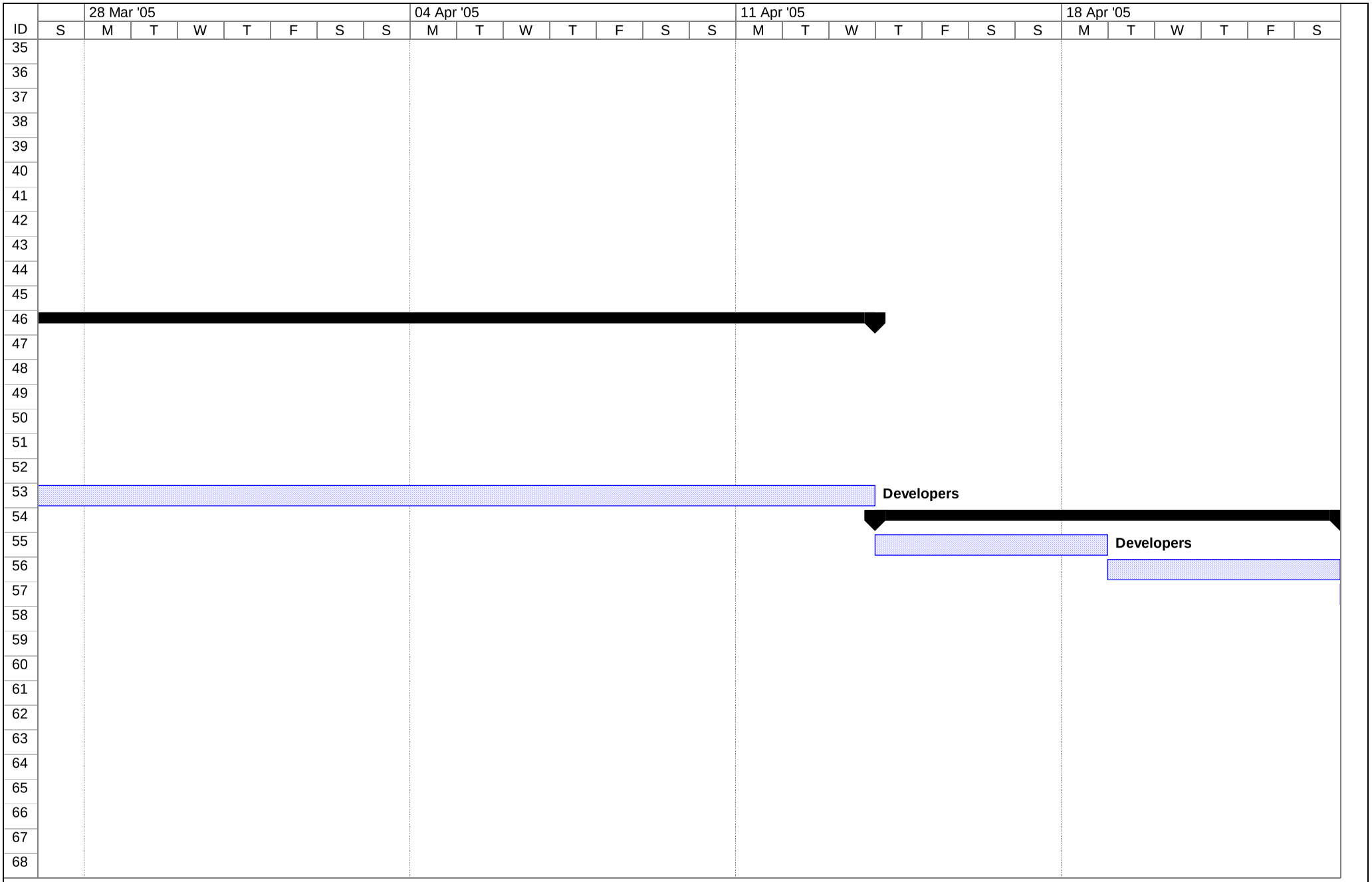

ID
06 Dec '04
13 Dec '04
20 Dec '04
27 Dec '04
S M T W T F S S M T W T F S S M T W T F S S M T W T F S
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
ors;Developers
Revisors;Developers
Revisors;Developers

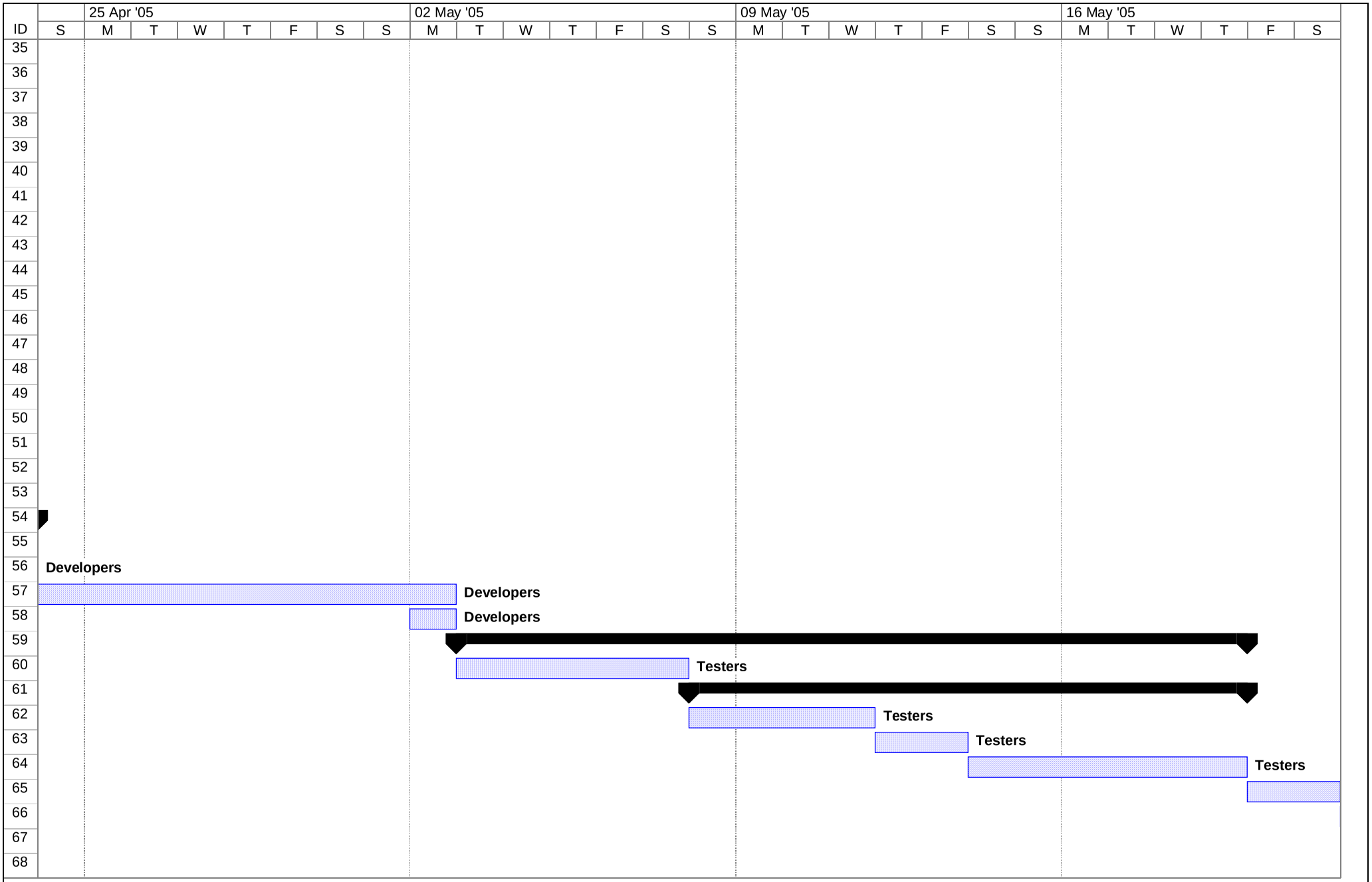

ID
06 Dec '04
13 Dec '04
20 Dec '04
27 Dec '04
S M T W T F S S M T W T F S S M T W T F S S M T W T F S
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
Presentation Team

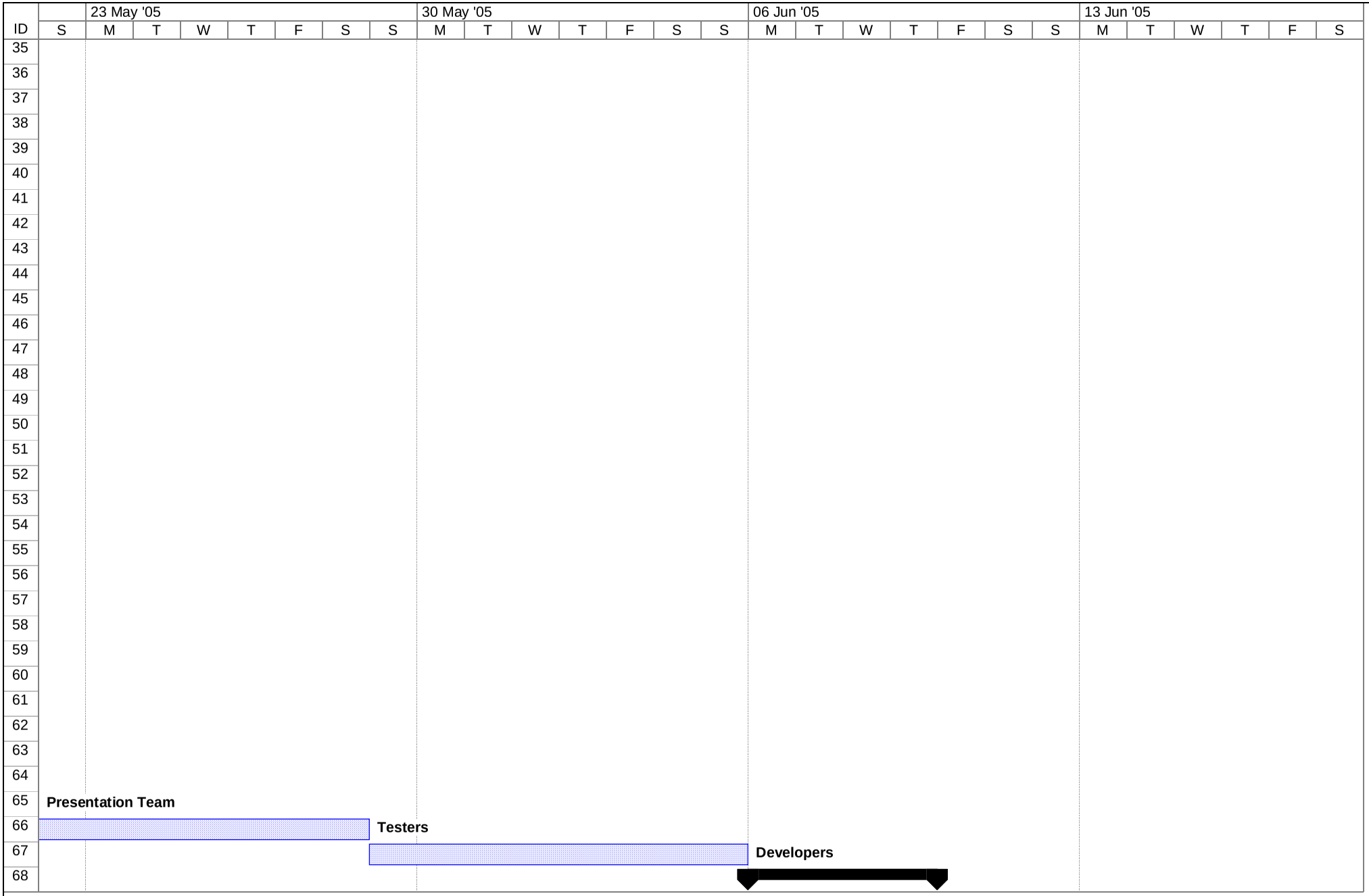

ID
03 Jan '05
10 Jan '05
17 Jan '05
24 Jan '05
S M T W T F S S M T W T F S S M T W T F S S M T W T F S
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
Developers
Reporters
Developers;Project Manager
Developers
Presentation Team


ID
31 Jan '05
07 Feb '05
14 Feb '05
21 Feb '05
S M T W T F S S M T W T F S S M T W T F S S M T W T F S
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
Devel
Devel


ID
28 Feb '05
07 Mar '05
14 Mar '05
21 Mar '05
S M T W T F S S M T W T F S S M T W T F S S M T W T F S
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
lopers
lopers
Developers
Developers
Developers
Developers
Developers
Developers


ID
28 Mar '05
04 Apr '05
11 Apr '05
18 Apr '05
S M T W T F S S M T W T F S S M T W T F S S M T W T F S
35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68
Developers
Developers


ID
25 Apr '05
02 May '05
09 May '05
16 May '05
S M T W T F S S M T W T F S S M T W T F S S M T W T F S
35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68
Developers
Developers
Developers
Testers
Testers
Testers
Testers


23 May '05
30 May '05
06 Jun '05
13 Jun '05
ID
S M T W T F S S M T W T F S S M T W T F S S M T W T F S
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
Presentation Team
Testers
Developers

ID		23 May '05							30 May '05							06 Jun '05							13 Jun '05					
	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S
69																												
70																												
71																												

Technical Communicators

Technical Communicators

Presentation Team


Project: Software Development
Date: Mon 10.01.05
Task
Split
Progress
Milestone
Summary
Project Summary
External Tasks
External Milestone
Deadline
Page 15