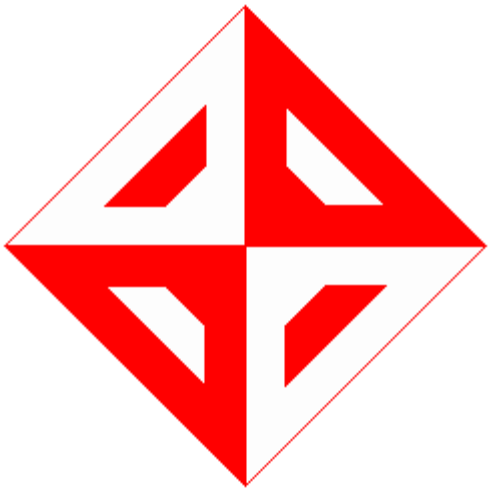


MIDDLE EAST TECHNICAL UNIVERSITY COMPUTER ENGINEERING DEPARTMENT

CLIMB PLANNER

CENG 491 INITIAL DESIGN REPORT

aiDSoft

TEAM NAME: AIDSOFT

PROJECT: CLIMBPLANNER

PROJECT ASSISTANT: ALİ ORKAN BAYER

PROJECT ENDORSER: ASELSAN

PREPARED BY:

İsmail PARLAK 1395409

Mustafa DİKİCİ 1394956

Ozan KEÇECİOĞLU 1395185

Yetkin SAKAL 1395425

TABLE OF CONTENTS

INTRODUCTION

Motivation

As a group of four senior computer engineering students, the AidSoft team aims to build a graphical tool to facilitate the analysis and preparation process of mountaineering clubs & societies.

Classified as a professional sports branch, a hobby or a profession, mountaineering embraces the aim to reach the top of a mountain regardless of its height. It is also noteworthy that the type of land the climbing action is performed varies remarkably as a result of different land & weather conditions. Hence, the discipline must be capable of overcoming the potential risks at analysis period in which all the precautions for the climb to be successful are taken.

The AidSoft team considers it of utmost importance to come up with a robust solution for mountaineers to input the conditions on which their climbing plan depends. The expected output of the software would then be an estimate on how long the climb will take and a number of potential climb paths where camping is available.

Project definition

As the project name suggests, Climb Planner aims to provide the users with the ability to prepare their custom climbing environment and view the climbing path in a 3D visualization scheme.

In order for the mountaineers to adopt the system of analyzing a specific climbing path, the software possesses a number of instructions to help the users do exactly what they want. In addition, the users are encouraged to navigate through the user interface to see what options they have and how wide the list of available input parameters they can add is. This would be a key factor in benefiting from the software as much as one can.

The system is made up of modular components to allow new maps to be added into the program. This feature not only enhances the capabilities of the software but also

provides the users with the option to create custom maps and examine the subtle differences between the climbing paths generated.

GRAPHICAL USER INTERFACE

Introduction

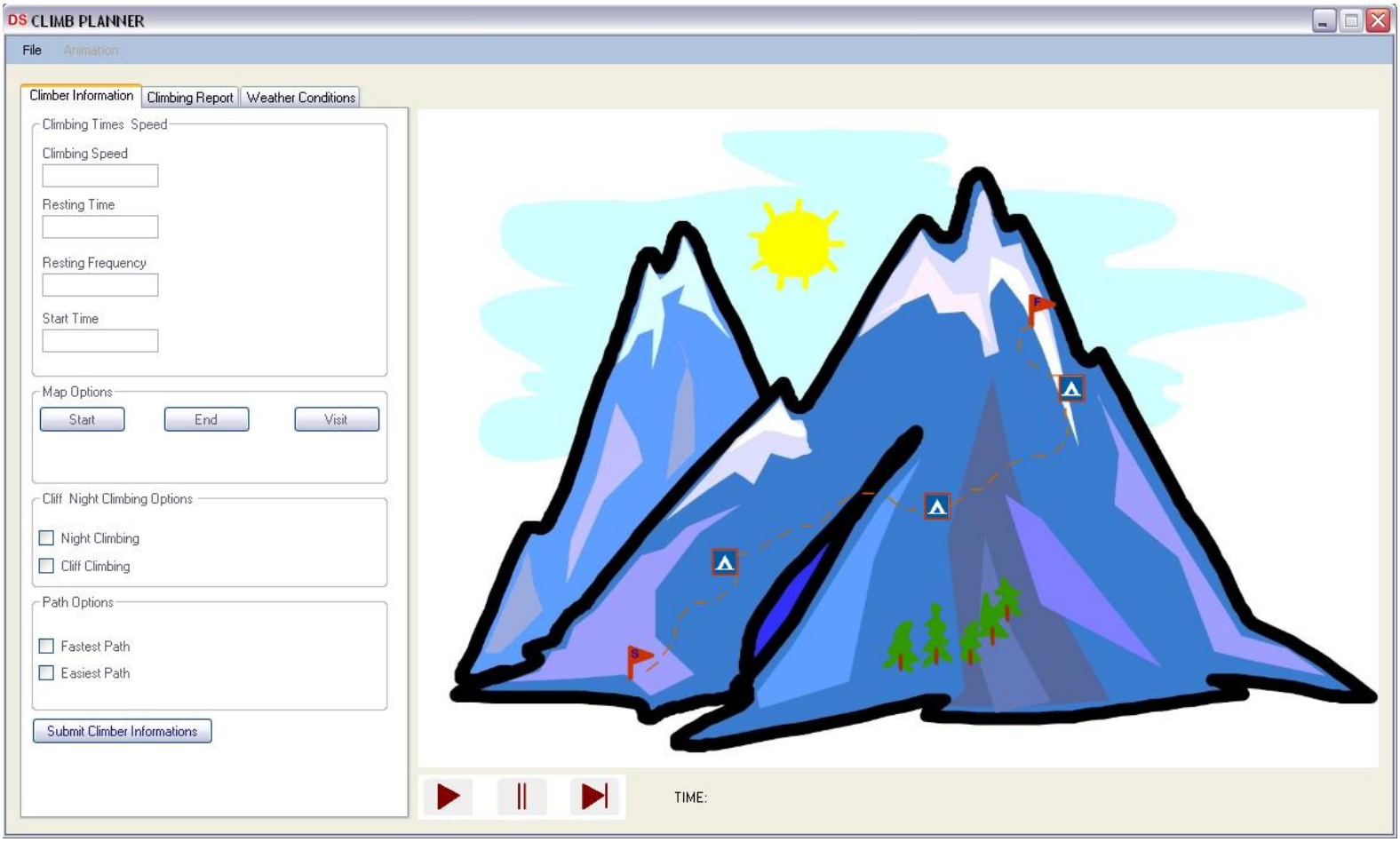
Climb Planner software helps mountaineers to plan, simulate and animate their climbing session. Since not every climber is a computer expert, GUI (graphical user interface) must be easy to manage so that mountaineers can easily use the software.

Below, we describe the GIU.

Main Window

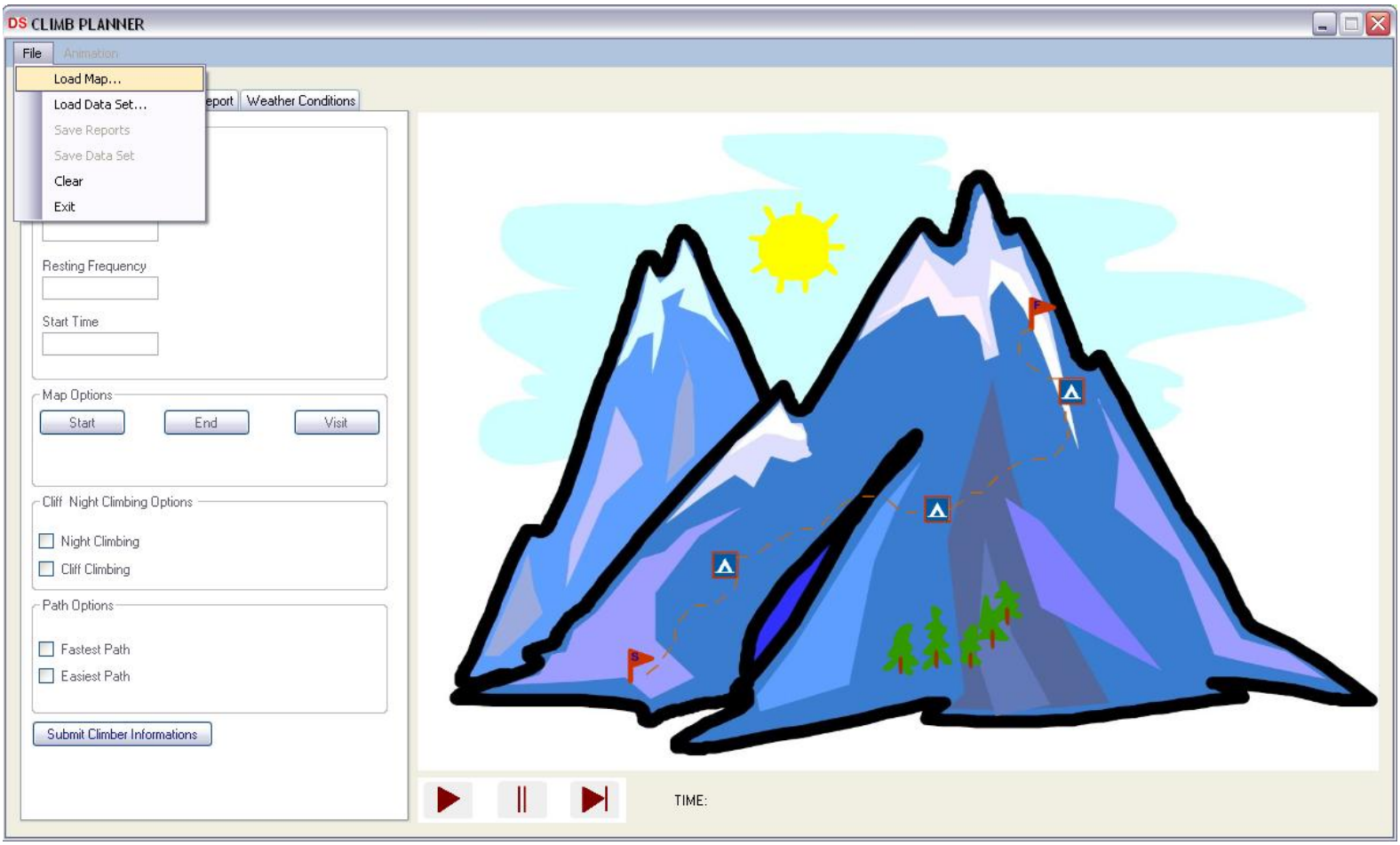
This is the main window of Climb Planner. It contains:

- File and Animation menus
- Climber Information, Weather Conditions and Climbing Report tabs
- Play, Pause and Replay buttons
- Animation viewport



These features will be explained below.

File Menu



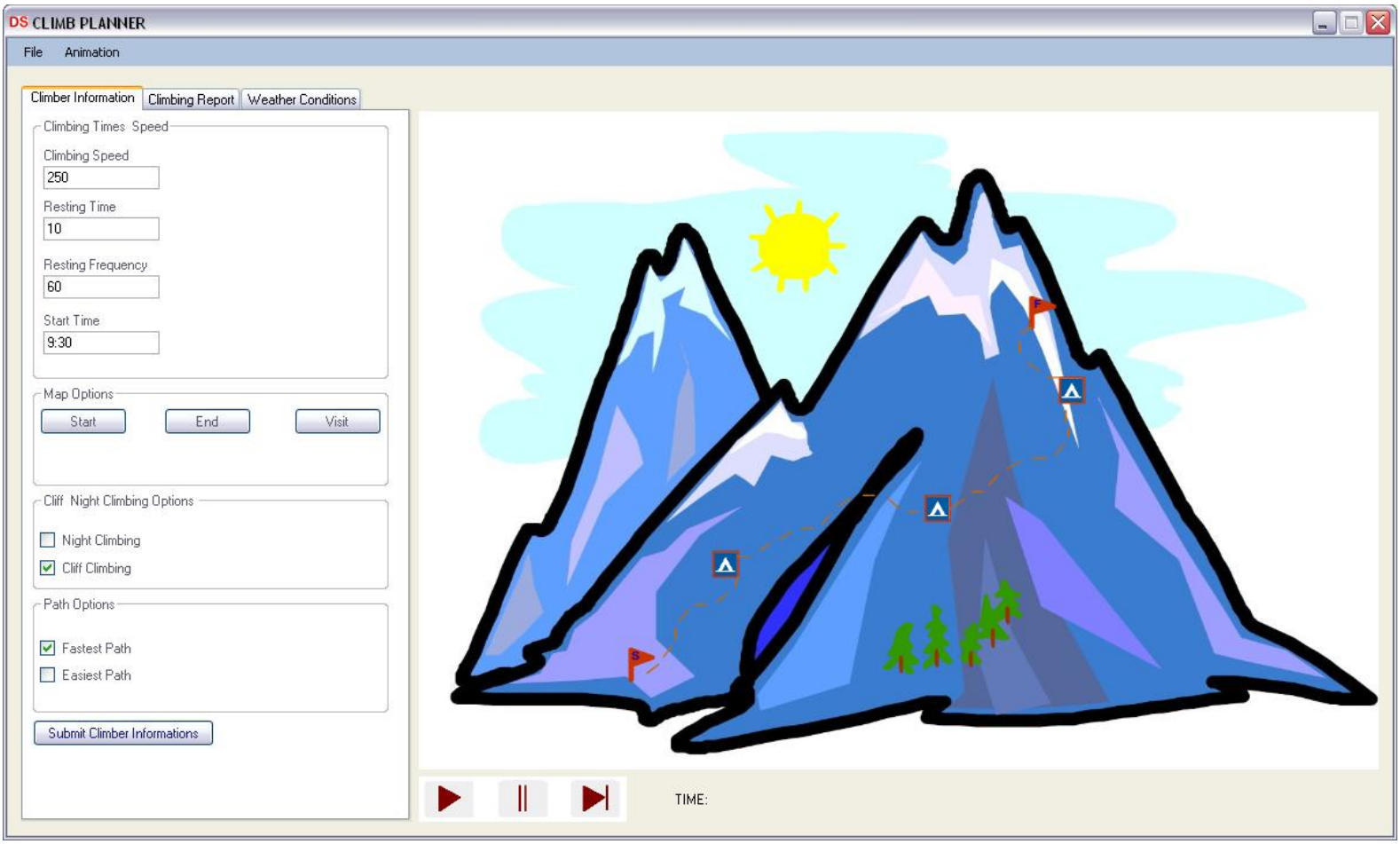
When Climb Planner is first run, only File menu is activated. In case of clicking on anywhere on main window but File menu or minimize and close buttons, Climb Planner will maintain its initial state. File Menu contains the following buttons:

- Load Map…
- Load Data Set…
- Save Reports
- Save Data Set
- Clear
- Exit

Save Reports and Save Data Set buttons are initially inactive. By activating only "Load

Map…", "Load Data Set…", "Clear" and "Exit" buttons we force user to load the map(s) or a data set. After loading map(s) or data set, Climber Information tab will be active. By clicking on "Load Map…" user can load a map. After loading height map user again can load any vector map(s) such as rivers or cliffs if available.

Climber Information Tab



This tab is required to collect mountaineers' skills and their plans for their climbing activity.

Climber Information tab contains:

- Climbing Speed
- Resting Time
- Resting Frequency
- Start Time

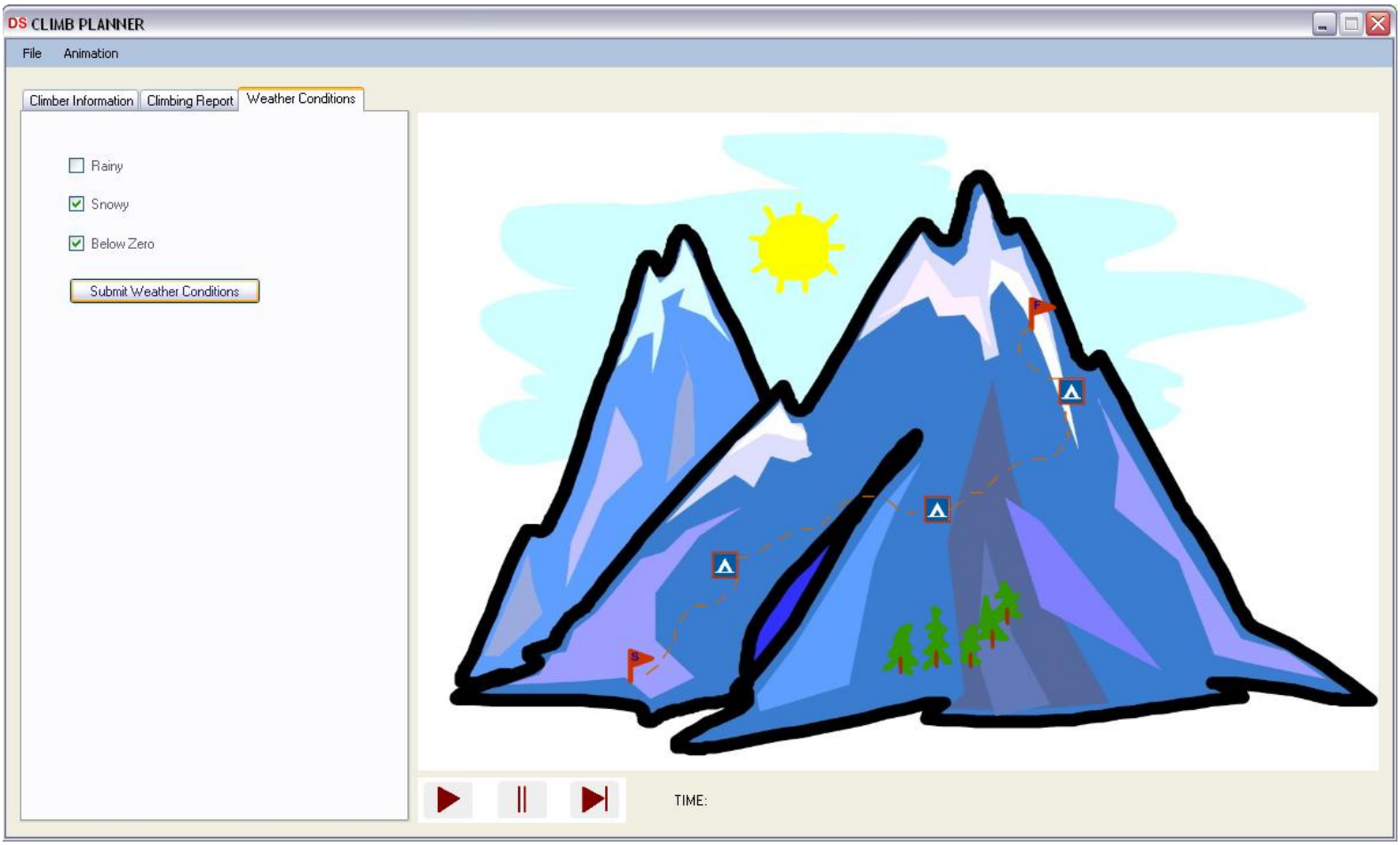
  Textboxes,
- Night Climbing
- Cliff Climbing

- Fastest Path
- Easiest Path

  Check boxes,
- Start
- End
- Visit Buttons

  By clicking on start button user can click on the map on viewport for pointing starting point of climbing activity. Similarly by hitting End button user is able to click on height map to point ending spot of climbing activity. If the user wants to visit certain spots on mountain, s/he can click on visit button and then click on desired positions on mountain map.

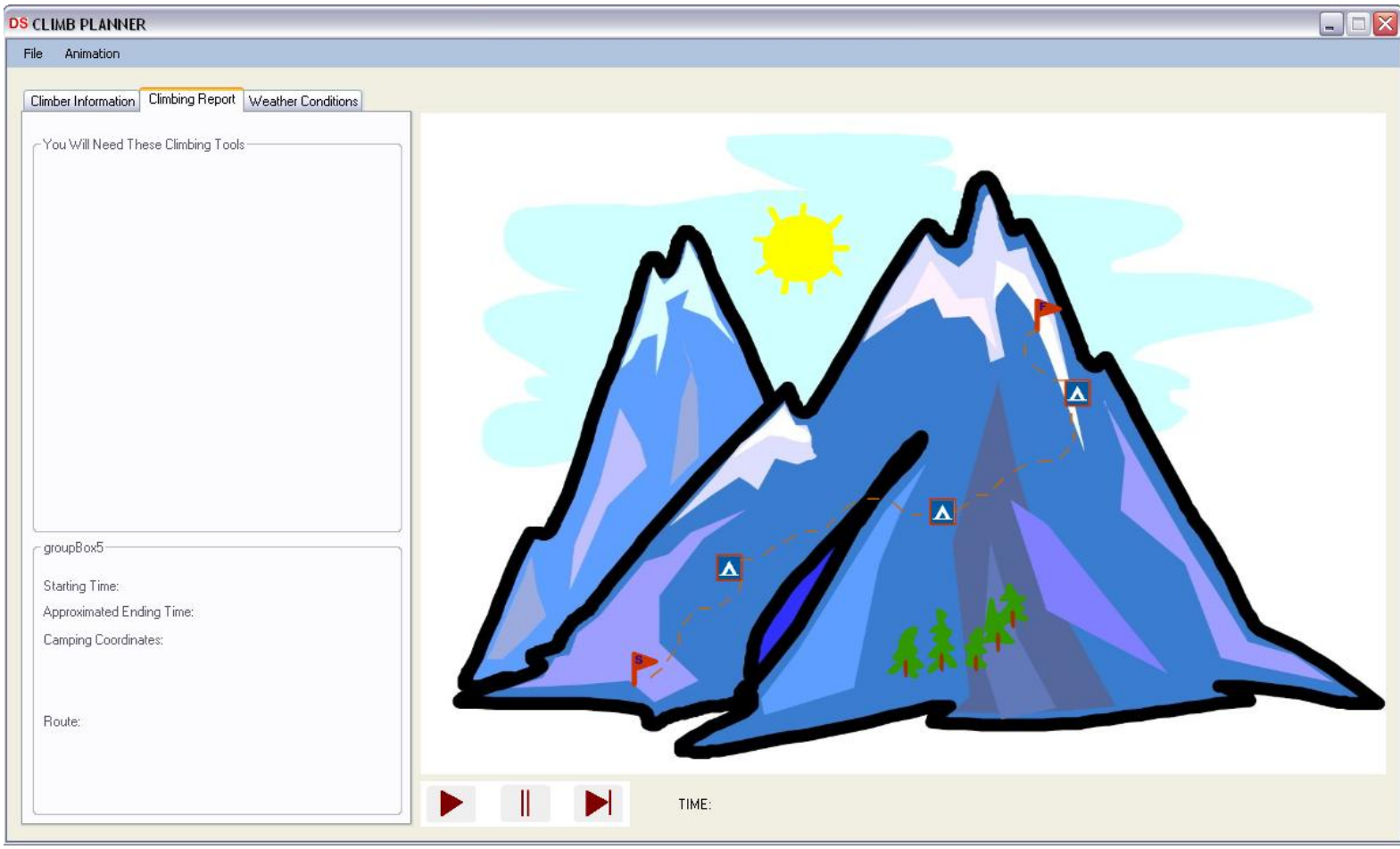
Weather Conditions Tab



This tab is optional. User does not have to check any of the Rainy, Snowy or Below Zero checkboxes. But, if the user has any meteorological information about the desired mountain, s/he can check one or more boxes and click on “Submit Weather Conditions” button.

This tab might be improved later by adding more detailed meteorological condition options.

Climbing Report Tab



In this tab we inform user about their climbing plan which is calculated and deducted by AI. This tab is not activated before submitting climber information.

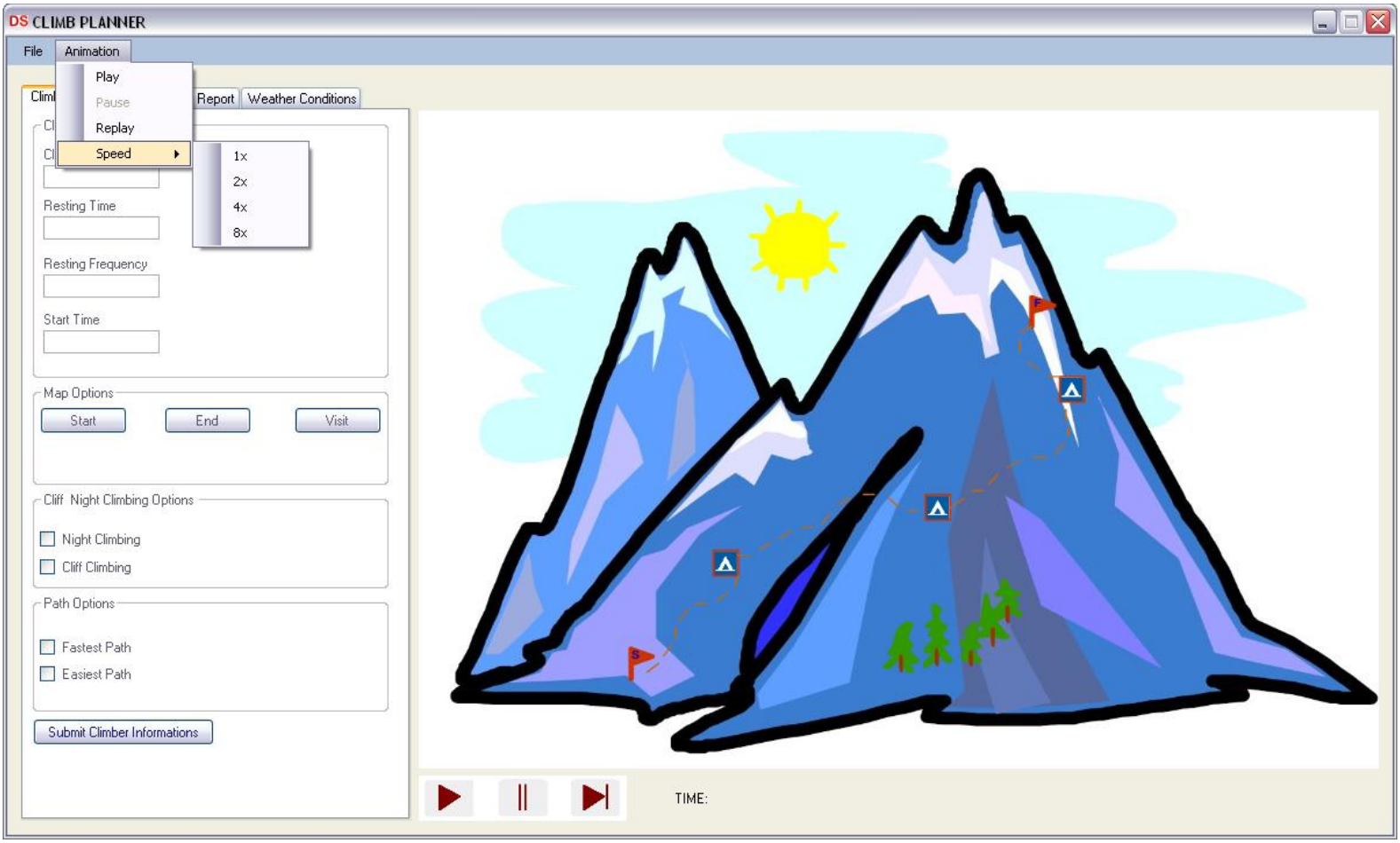
This tab contains “You Will Need These Climbing Tools” and “Climb Plan” group boxes.

In “You Will Need These Climbing Tools” section, climber will be informed about the climbing tools which are necessary to handle climber’s mountaineering activity.

In “Climb Plan” section, user will be informed about the starting time and approximated ending time of climbing activity. Moreover, user will also be informed

about suitable camping places' coordinates and the coordinates of the route to be followed.

Animation Menu

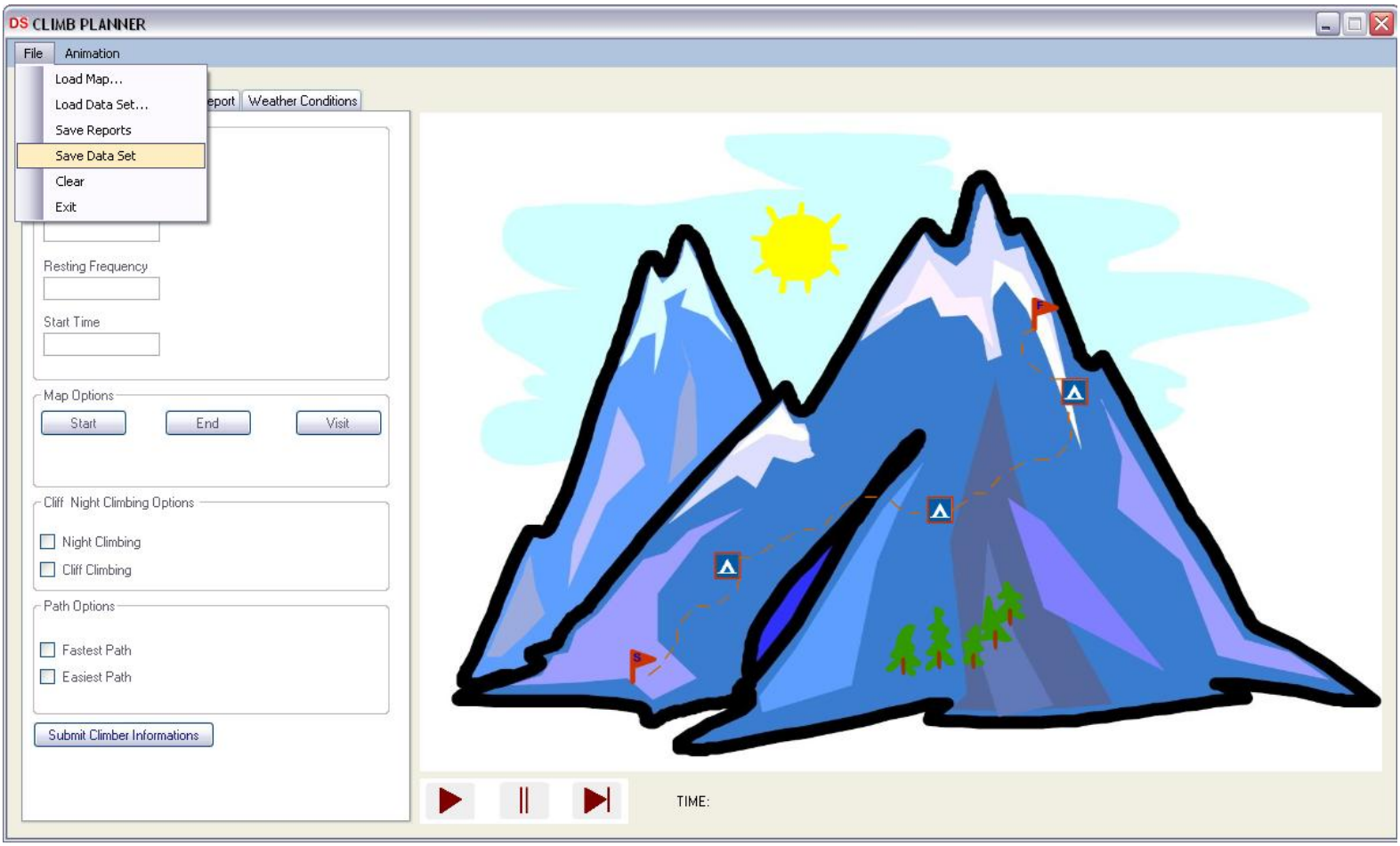


This menu is not activated before submitting climber information. In addition to speed selection, animation menu has the same features as the buttons at the down-middle side of the main window, which are Play, Pause and Replay. User is able to select the animation speed (1x, 2x, 4x, 8x). After selecting the animation speed, Play, Pause and Replay buttons are activated. Once user clicks on "Play", animation of the calculated climbing is shown on the right hand side of the GUI.

Picture on animation viewport here is obviously caricaturized just to show where the animation will take place. Actual animation will be rotatable and scalable by using mouse and it also will illustrate movement of the sun, starting, ending points of the route, land type, calculated route, North, South, East, West directions and camping points.

If the user clicks on pause button during animation, animation will be paused. In clicking replay case, the animation will start from the beginning.

File Menu (After Submitting Climber Information)



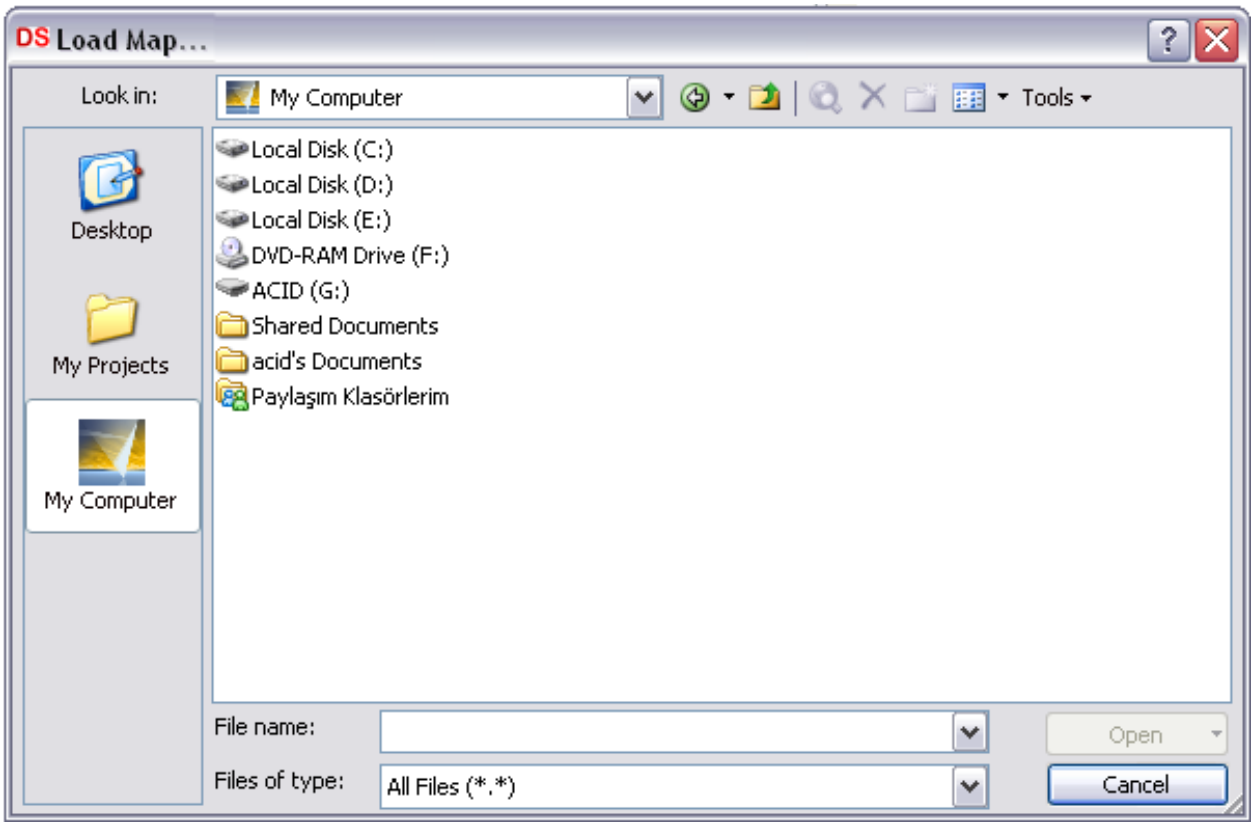
After user submits climber information, “Save Reports” and “Save Data Set “options in File menu will be activated. By clicking on “Save Reports”, user will be able to

select the destination for saving the complete climbing plan in text format. By hitting “Save Data Set “, user will be able to save all previously loaded maps (Height map, vector maps) as a single data set. Saving all maps as a single data set eases the load of same maps for future use. If user wants to use the same maps in future, s/he can load all of them by just hitting “Load Data Set” and then select previously saved data set.

If user clicks on “Clear”, main window will return to its very first initial state.

When user is done with Climb Planner, by clicking on “Exit”, s/he can terminate the program.

Load Map Window

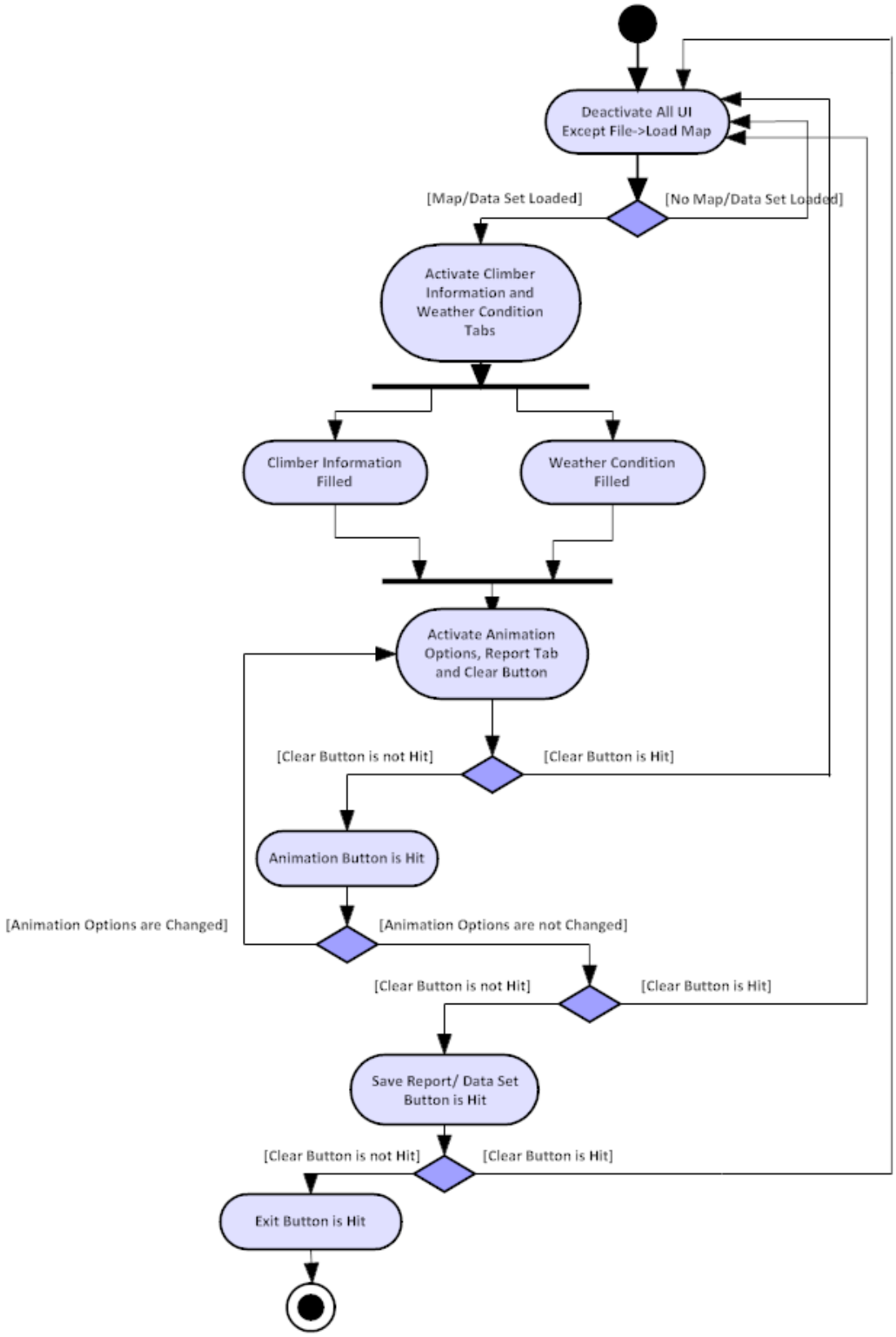


“Load Map…” window is a usual Windows application load window. On this window shortcut to desktop, my computer and my projects (data sets) are available. User is also able to use the drop down search menu which is located on the top most side of the load map window.

“Load Data Set…” window is as same as this window except the title.

This window will be changed later for most convenient use.

GUI Activity Diagram



ARCHITECTURAL DESIGN

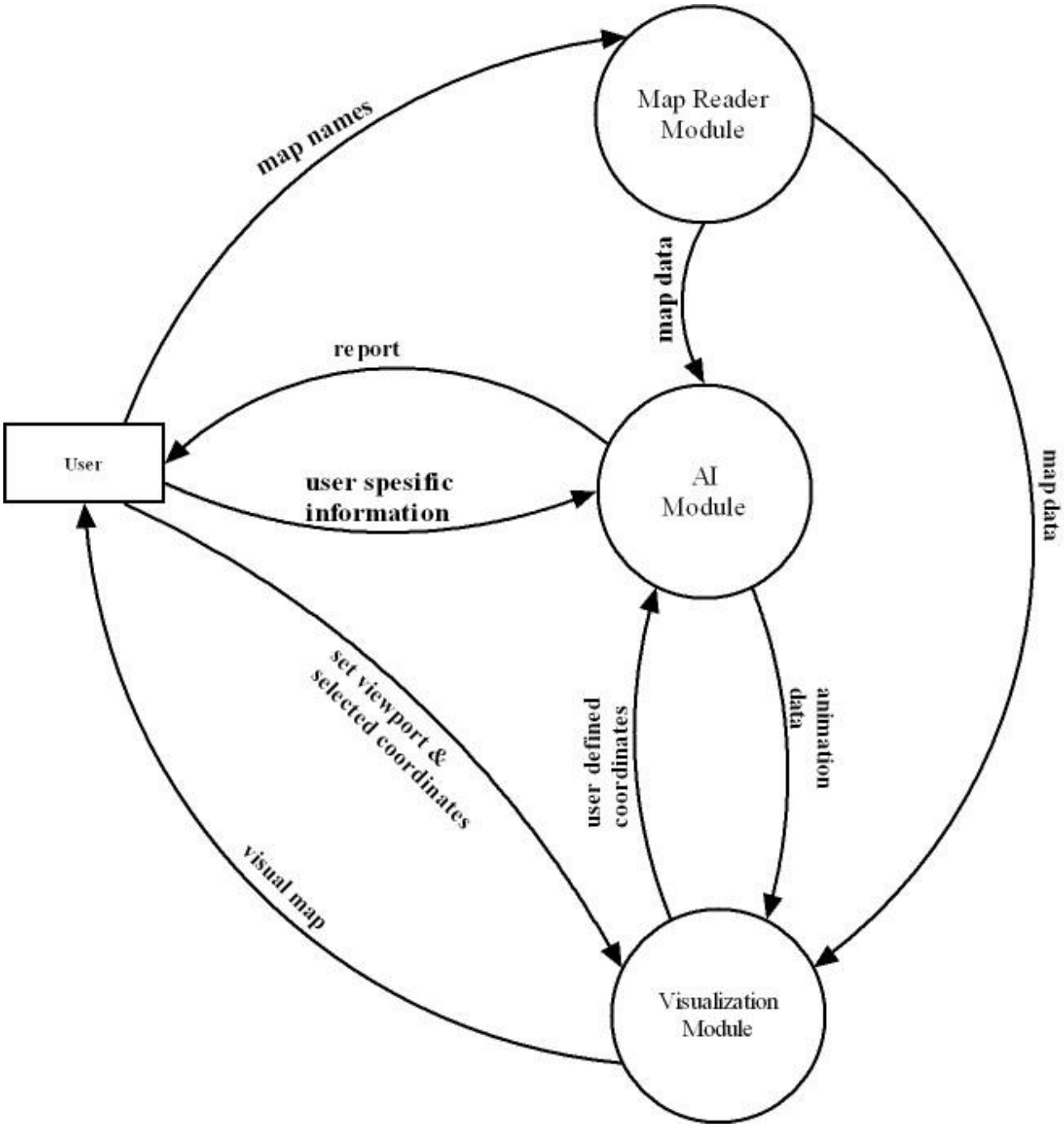
In this section, reader will be informed about Climb Planner's architectural design, first by a general perspective, afterwards by a detailed description of sub components of the software. It should be noted that anything written below may be subject to a change or modification according to feedbacks, since this report is not the final version of the software design.

Designing Climb Planner's architecture, our main goal has been to maintain simplicity and extendibility. Since our team is inexperienced and the time that team have or can spend to this project is limited, any serious conflict about the design, whenever it is occurred - in implementation or in the further progress of the design -, is likely to cause a failure of this project . At this point, determination of the general borders of any components and their abilities becomes critical. This is why a minimalistic approach is used. If the design seems to be narrow or insufficient, reader should consider this fact and must remember that whenever it is possible, the scope of the design will be extended to include wider abilities.

Main Design

This part will be an overview of the software. It can be thought as a review of the one in the requirement analysis report mostly, but this version includes the modifications and decisions that are made through the interval.

Climb planner is composed of four main parts to keep the minimal requirements of a possible climbing simulation. These are a simple and clear GUI, a 3D visualization module, an AI module to analyze and process the climbing route, a map reader module to maintain the environment data to the visualization and AI modules. The roles of these modules are defined to satisfy a strict interface through the data flow inside the application. Below, data flow diagram level 0 describes the communication between modules where GUI is represented as user.



The data flow and modules are described below in a general manner.

Information taken from the user by GUI is divided into 4 main groups which form the all data that application will need (except the raw map data). These are described below:

Map names: User selects which mountain map is to be simulated. These maps are sent from GUI to map reader module as operating system dependent logical file locations to be opened and handled directly.

User specific information: Environment and climbing requirements gathered by GUI is combined in this data flow group. Mainly, these are climbing speed, resting time, resting frequency, weather conditions, starting time, night climbing choice, diff climbing choice and desired path type.

Viewport information: Since the user will be allowed to wander inside the 3D environment, position and orientation information of the viewing camera will be gathered by mouse and keyboard inputs and sent to the visualization module to be handled in real time. Visualization module will process the viewport according to these inputs.

Selected coordinates: This data group represents the information of coordinates which user wants to be processed through the simulation. These are starting and ending points of the mountaineering activity and desired fixed visiting coordinates.

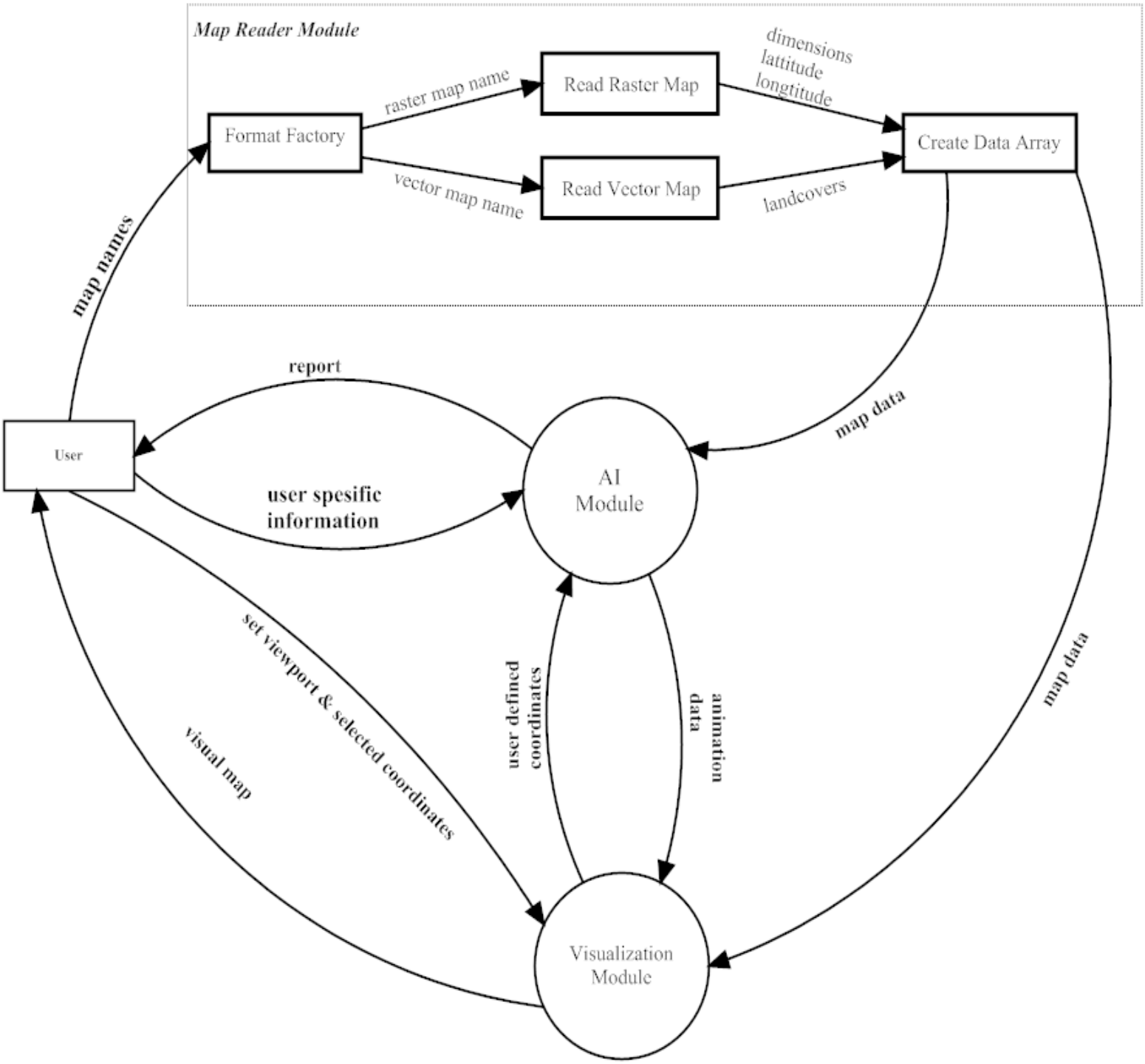
This process is handled by visualization module to unify the screen output and AI output since it would not be nice if the location of data is not the same with which user sees. Visualization module will translate these mouse data to real coordinates, display them in screen and send them directly to the AI module.

Visualization module will be responsible of building the 3D representation of the mountain using the 2.5D height map data, mainly. It constructs its own polygonal data and uses it through the simulation. Also module combines the user defined coordinates, which are mentioned above, with this polygonal data and animates the climbing route which it takes from AI module. More specific representation of the visualization module can be seen in DFD level 1.

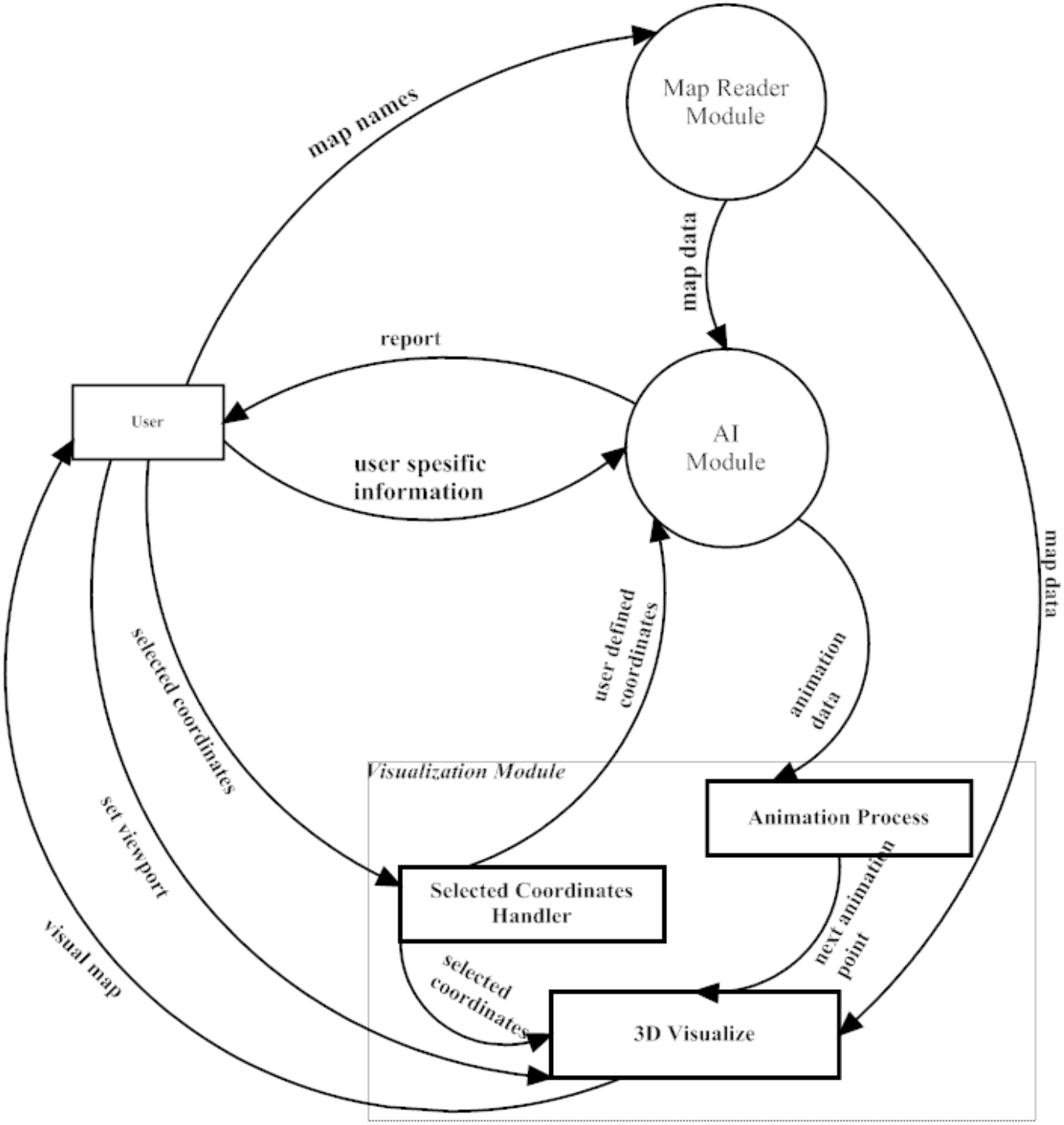
AI module abstracts its internal structure with the other parts of the application totally. It gathers user specific inputs as states to be used in the route determination of grid based map data. Its output is time-location-progress oriented animation data to visualization module.

The main role of the map reader module is to achieve an abstraction between GIS data formats and map data which will be used by AI and visualization modules. Simply it reads the raw map data (this is not shown in the dfd for the sake of convenience) and converts it to per-grid data. Doing this, it will make use of a common data structure - called **MapData** - which will be realized and used by the whole application without any further needs of transformation. Through the lifetime of every single simulation, this module is called for one time, at the beginning. It will be capable of calculating intergrid slopes, which will be needed by AI module in the determination of the elevation weighted shortest path. A birds-eye view of this module can be seen in DFD level 1, below.

MapReader Module Level1 Data Flow Diagram

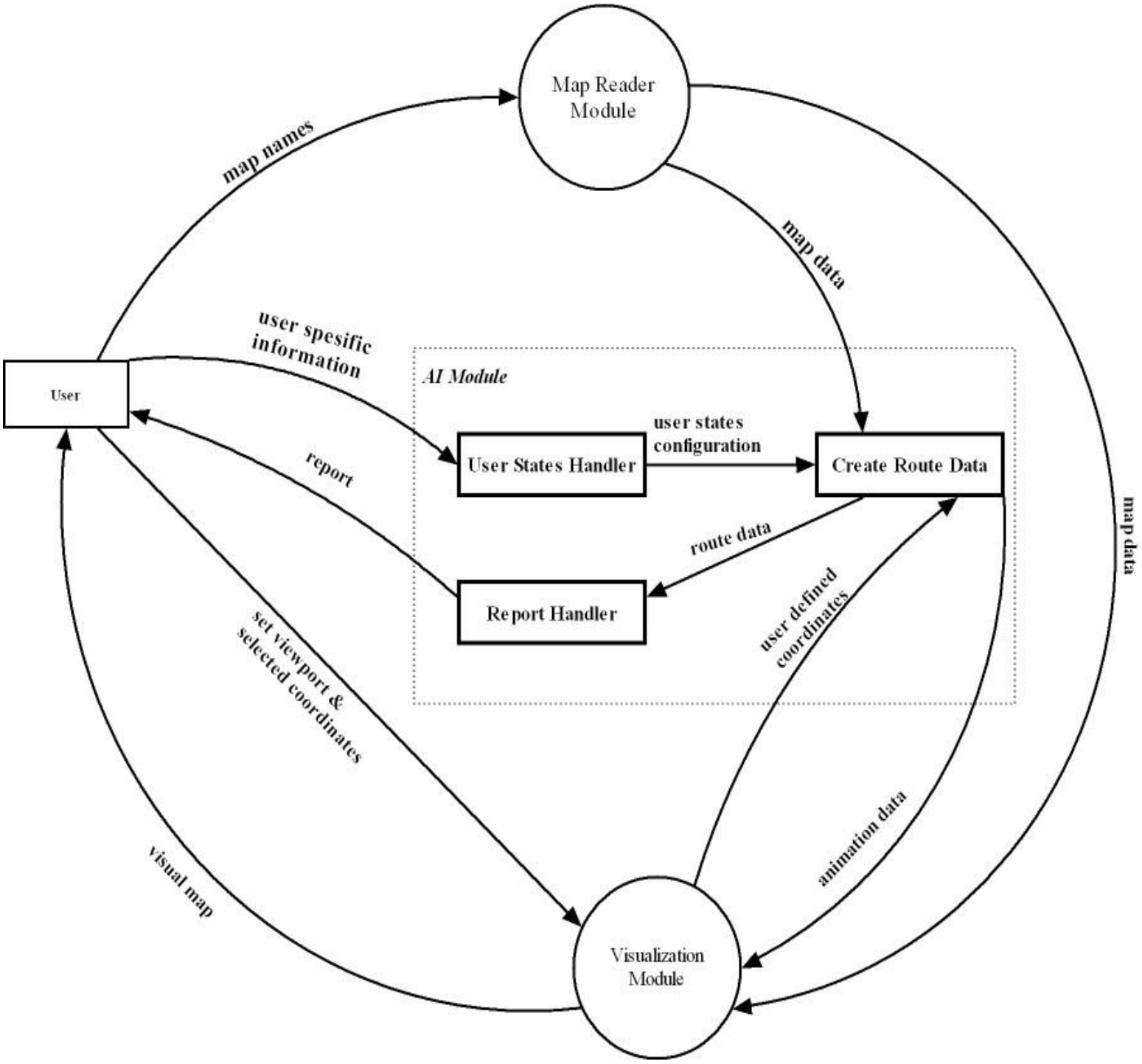


Visualization Module Level1 Data Flow Diagram



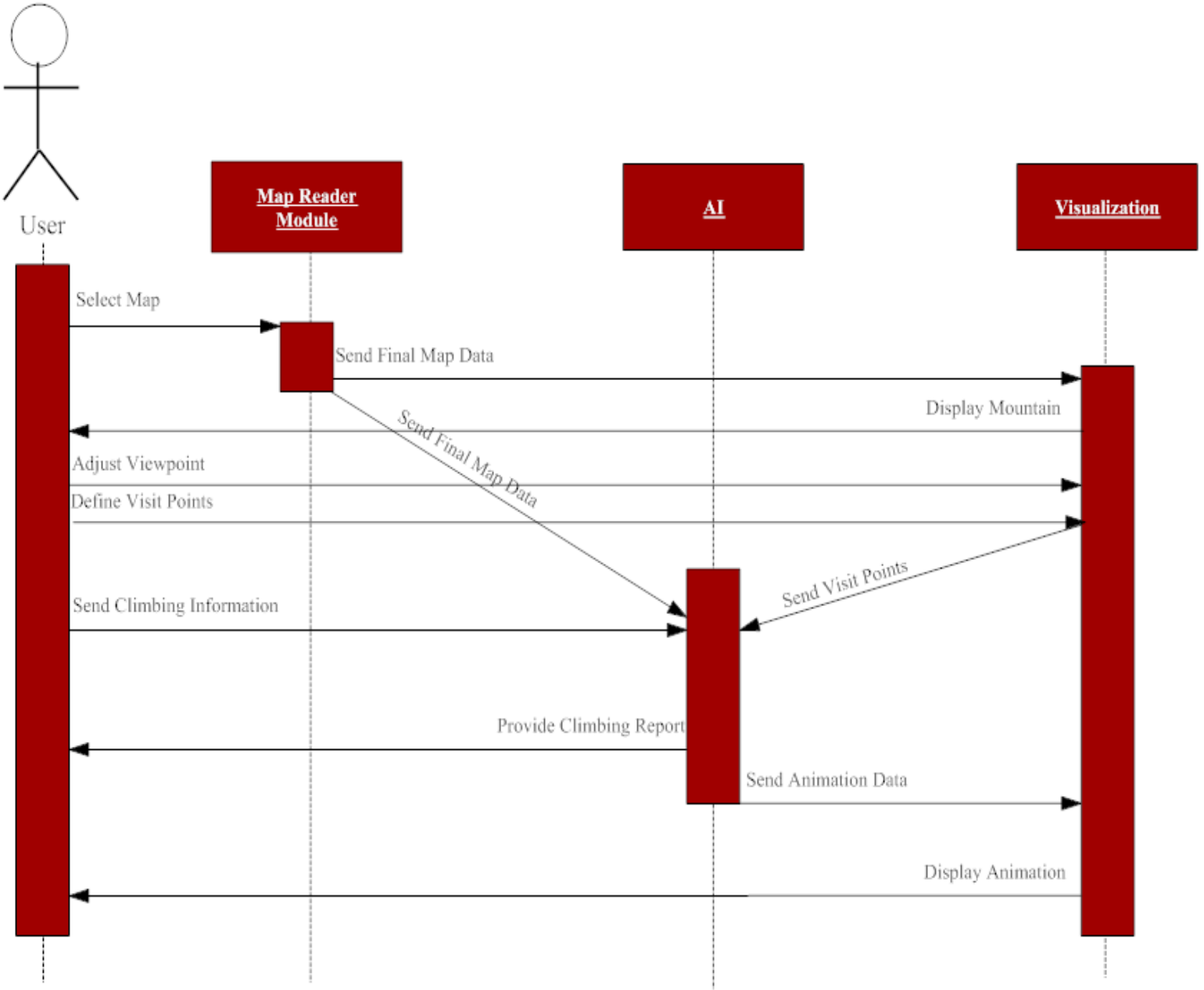
Visualization module has three functionalities called animation process, selected coordinate handler and 3D visualize. Animation process gets the animation data from AI module and gives the next animation point to 3D visualize. By getting map data, next animation point, selected coordinates and viewport settings, 3D visualize returns the visual map to user. Selected coordinates handler gets the desired coordinates from user and deliver them to 3D visualize and AI module.

AI Module Level1 Data Flow Diagram



AI module is made up of three functionalities which are user states handler, create route data and report handler. Create route data gets the map data from map reader module and creates the route data to deliver it to report handler. After report handler gets the route data, it delivers the related report to user. User states handler gets the user specific information from user and configures the user states to be delivered to create route data.

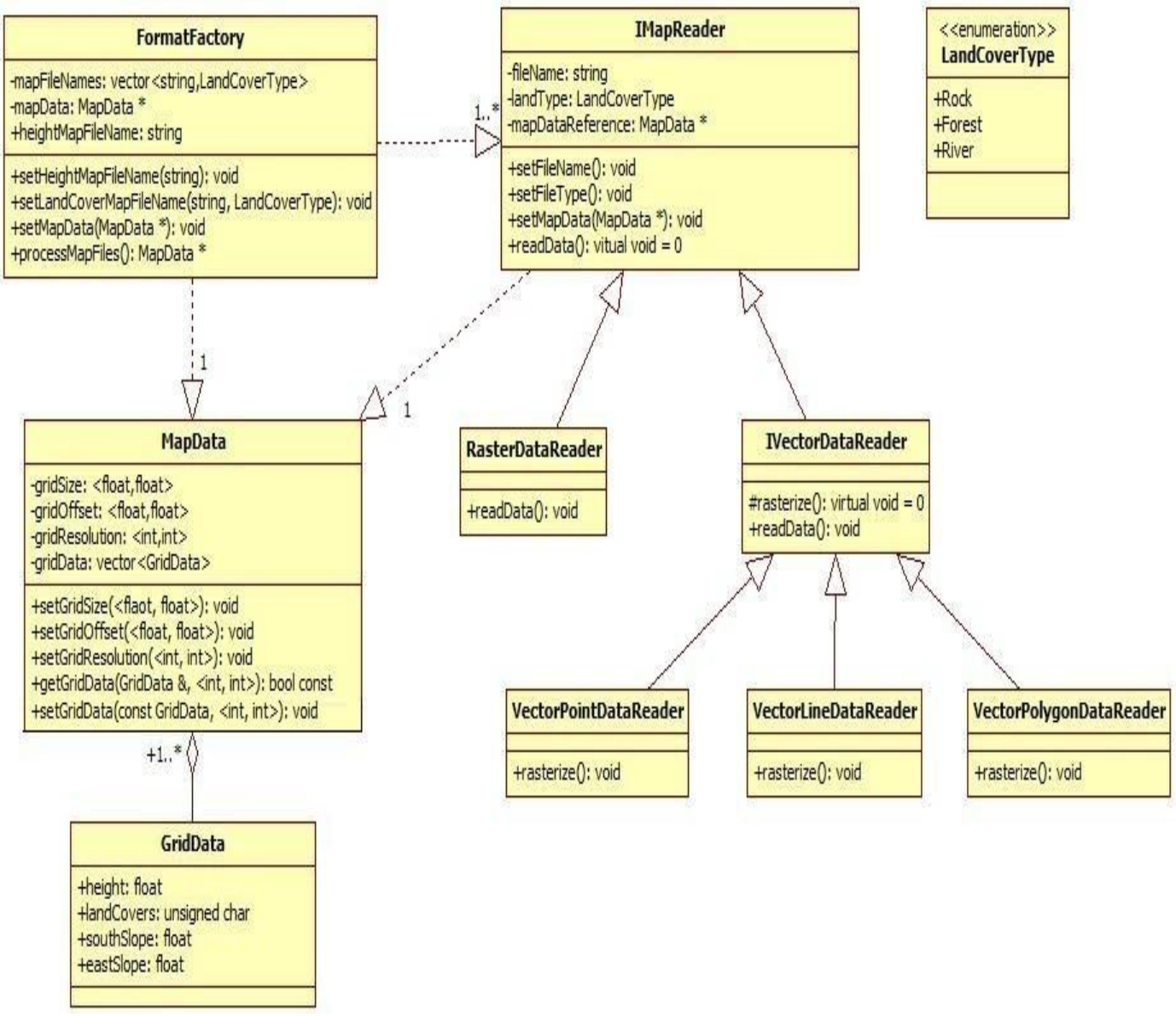
Main Sequence Diagram



User selects map(s) (or data set) from GUI File menu. Map reader module reads the map(s) and sends final map data to both visualization and AI modules. Once Visualization module gets the final map data, it begins to display the selected mountain to user via GUI viewport. Now user can adjust the viewport by mouse and define the visit points. Now the defined visit points go to visualization module and then to AI module. After AI module gets the user specific information (climbing information), it calculates the path and returns the climbing report to user and returns animation data to visualization module. After visualization module gets the animation data, it displays the animation to user.

4.1 MapReader Module

The main role of MapReader module is to get the mountain maps in several GIS formats, process them and present them in a common data structure to the use of the other parts of the application. There will be at least 1, at most 4 types of maps which will be used in the application. These are a heightmap (Possibly in raster format) which its existence is a must, a rocky terrain map, one map of the existing rivers and bridges and a map of forests. Class diagram of the module can be seen below.



MapData is the main data structure which will be used by all the modules through the application. Because using grids makes the implementation of the shortest path algorithm much easier and efficient, MapData has been designed to keep data in grids. So any possible vector formats should be rasterized and converted to grids. For the sake of simplicity, all maps should be processed in the same resolution by AI module. At this point heightmap data becomes important to keep the grid size and count unique since it will define the resolution of the MapData for the analysis (In case of a vector formatted heightmap is used, resolutions should be pre-defined. We will modify the design in this situation.).

MapData stores the environment information per grid in its GridData array MapData::gridData. GridData is composed of a height value, a hexadecimal group of LandCoverType values called GridData::landCovers, and its slope to its south and east neighbors. MapData instance will also keep its resolution, offset (in latitudes and longitudes) and grids' size (in latitude and longitude).

Because there are variety of GIS formats, format specific properties and projection types, the most critical design aspect of MapReader module is its abstraction of map data. To satisfy a single interface to this module, OOP design pattern “Factory” is used. Called FormatFactory, not only this class has the ability to recognize any pre-defined GIS format and it creates an appropriate IMapReader object to read the data, but also it can manage IMapReader objects to keep the data integrity and process the map data to calculate the intergrid slopes itself.

When a FormatFactory instance is created, it will be fed with the map file names which user wants to be simulated. FormatFactory keeps these file names in mapFileNames vector structure. When all the file names are ready, (User may not define file names for all map types, in this situation a default terrain type will be used for that map type.)

FormatFactory::processMapFiles() is called to start a transformation between any possible GIS formats to MapData grid structure. FormatFactory realizes the formats of the given maps and creates appropriate IMapReader objects, which will handle the

reading and ,if the map is in vector format, rasterizing process of the maps. Firstly the heightmap should be translated to define the resolution of the MapData. After all the conversion, FormatFactory will calculate the intergrid slopes and return a complete MapData object that is ready to be analyzed and visualized.

In the implementation phase, MapReader will make use of OGR and GDAL libraries heavily to translate GIS formats. These libraries present a well defined abstraction of projection and GIS formats.

VISUALIZATION MODULE

Visualization is the key factor to realism. Presenting a realistic environment to the user will increase the attraction of a simulation software. But in the other hand, 3D visualization may be the most expensive component in the means of system resources. In the design of this module, balance between a realistic and system friendly approach is the most critical phenomenon.

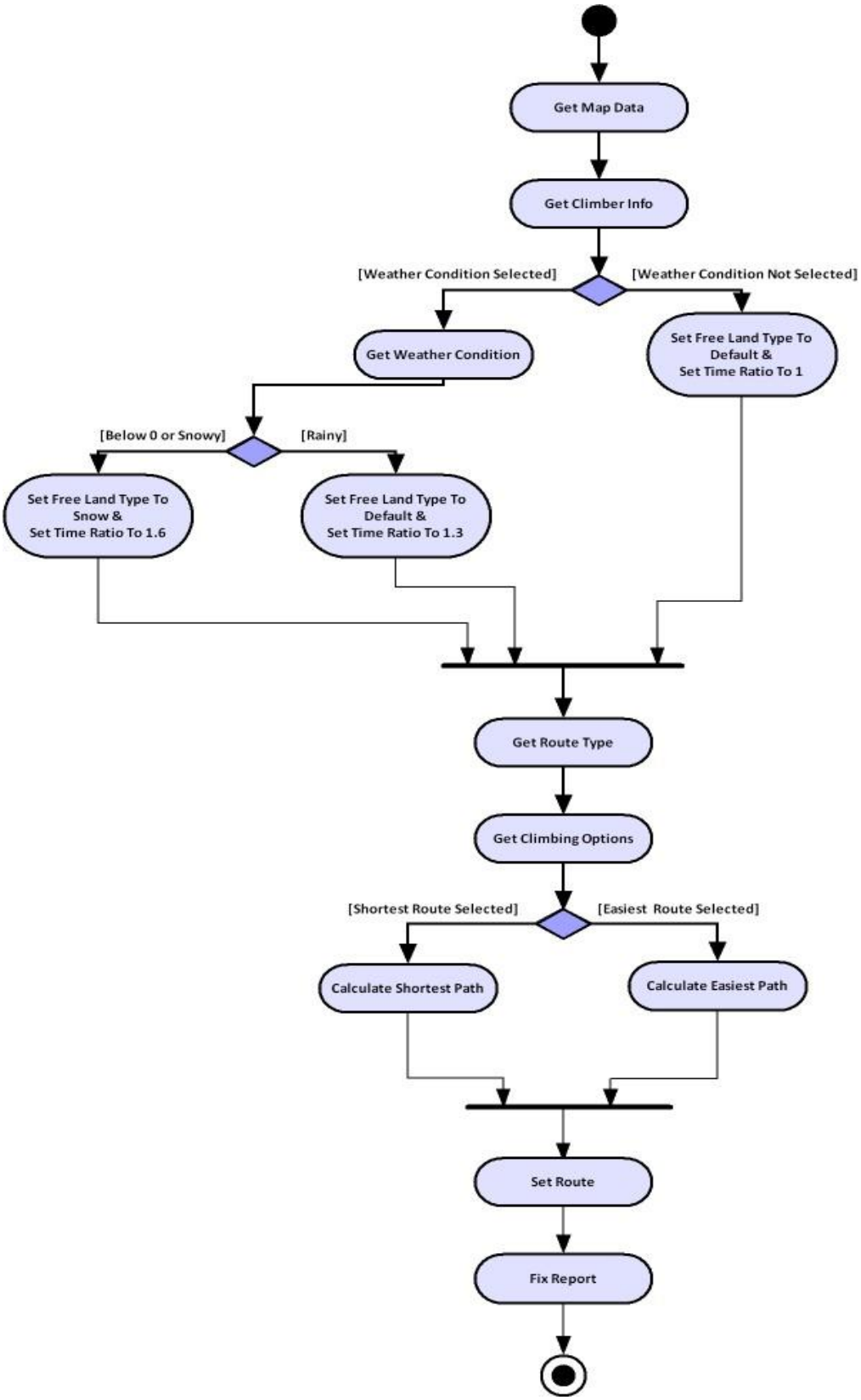
The main environment will be maintained from the heightmap, where other maps will constitute the terrain textures of the environment. It is possible that 2D modeled sprites can be used to simulate the forests and mountaineers through the animation. For a realistic image, sun light will be simulated. Sun will change position through the animation to indicate the time of day.

When the existing techniques are considered, octree algorithm is the most successful implementation of open area simulation. Using octrees, the unseen part of the simulation environment can be estimated roughly. Drawing only the visible part of the environment boosts the performance. So when the visualization module is initialized, an octree of the environment will be built first.

Another rendering technique of open area simulations is the use of level of detailed models. By level of detailed models, when an object is far from the camera, it is rendered less detailed considering its original level of detail. This method can be implemented to the heightmap terrain, where far grids to the viewing camera are combined to compose a larger grid, which means less calculation for rendering. Octree will help to implement this method by calculating the distance of the camera to the octree nodes.

OpenGL will be used in the visualization module. Moreover, it is highly considered that Ogre, a open source game engine, may be used for the visualization module. Ogre supports many algorithms, including octree and lod (level of detail) methods, for open area simulations. Since it is possible that GIS data are largely in size, it is likely that heightmap will be costly to simulate. Visualization module design will be formed in detail after experiencing with the implementation of such a heightmap and decision of using Ogre or not will made after these experiments.

Artificial Intelligence (AI) Activity Diagram



AI module takes both map data and user options as input and calculates the desired path, suitable camping regions' coordinates and it determines the climbing tools for future usage. In addition, AI calculates the time at any point on route after climbing begins and sets land type of the mountain for the animation process. If weather conditions are entered by user, AI does the timing calculations according to them by changing the time ratio number.

For shortest path calculation AI will be using A* (A star) shortest path algorithm. A* is a best-first, graph search algorithm that finds the least-cost path from a given initial node to one goal node (out of one or more possible goals). Considering the terrain we need to work on, A* is the most convenient shortest path algorithm for us.

For "easiest path", AI will only consider the inclination of the given mountain surfaces. The path leading to mountaineers' goal with the lowest over all inclination will be the easiest path.

TOOL RESEARCH

Graphical user interace (GUI) of the system will be created using the Qt library which complies with our requirements about the 3D visualization in OpenGL. Furthermore, it is a robust solution for this type applications where graphical user interaction gains importance. Independency of platform is another reason why Qt is preferred among the GUI libraries of the same kind.

Reading the maps, gleaning the optimum data sets from them and make the necessary transformations are among the main routines employed during the life cycle of the program. In order to efficiently complete these tasks, we decided to use GDL and OGR. This way, we will be able to perform the operations we need with no burden and as fast as possible.

Regarding the limited size of the maps used as inputs by the program, no database management system is incorporated into the software. It is also noteworthy that this approach decreases the general complexity of the system, reducing the number of read/write operations by a significant order.

PROTOTYPE IMPLEMENTATION

Prototype Approach

Since it is nearly impossible to successfully develop all of the features of Climb Planner software until the end of this term, prototype will be focused on some basic features of the software.

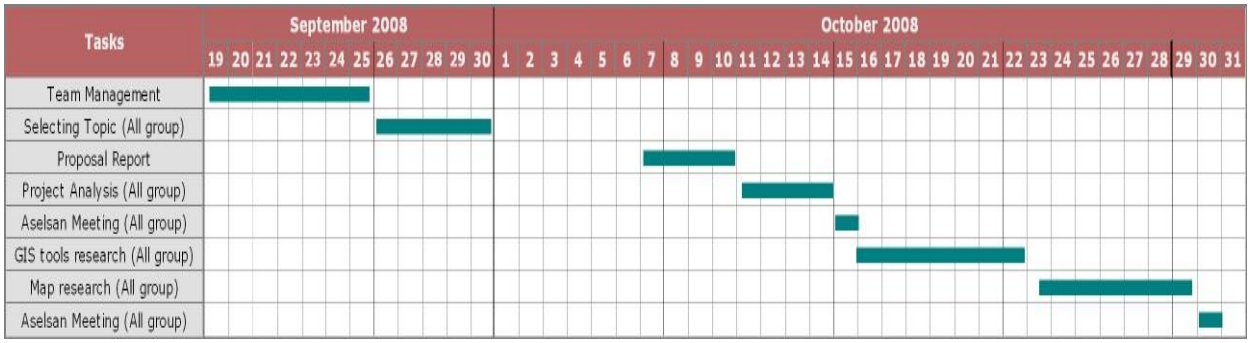
We aim to accomplish developing these features of the Climb Planner software until the end of this term:

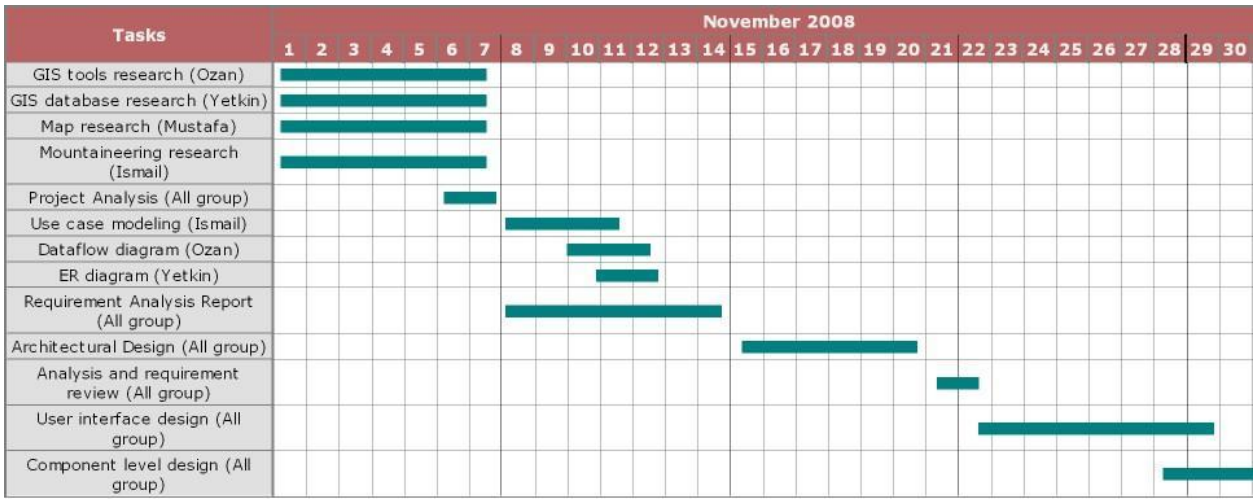
- By using OGR and GDAL we plan to at least read the available height maps of mountains and visualize some basic parts of animation
- What we refer by basic parts of animation are: animation will be in 3D, it will be rotatable and it will be scalable.
- We plan to fully develop the GUI

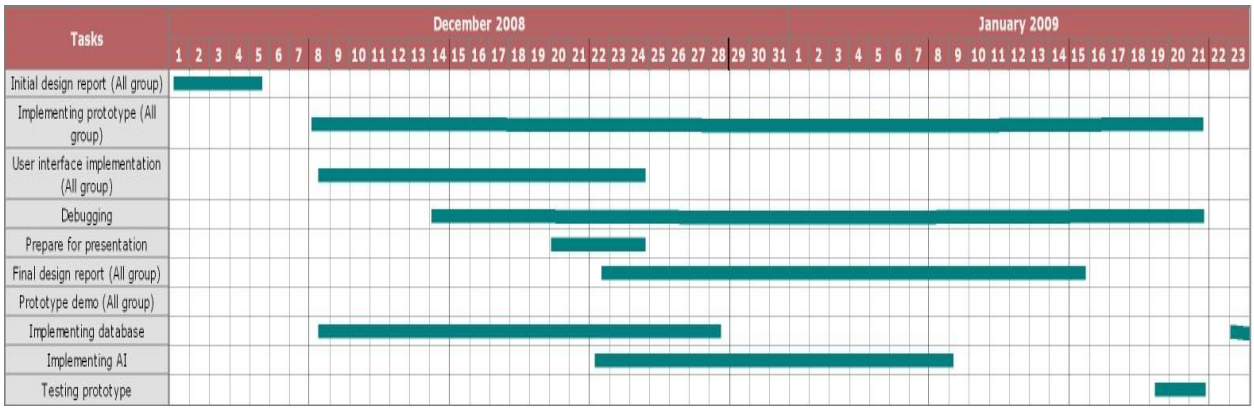
By developing these features we will accomplish these goals:

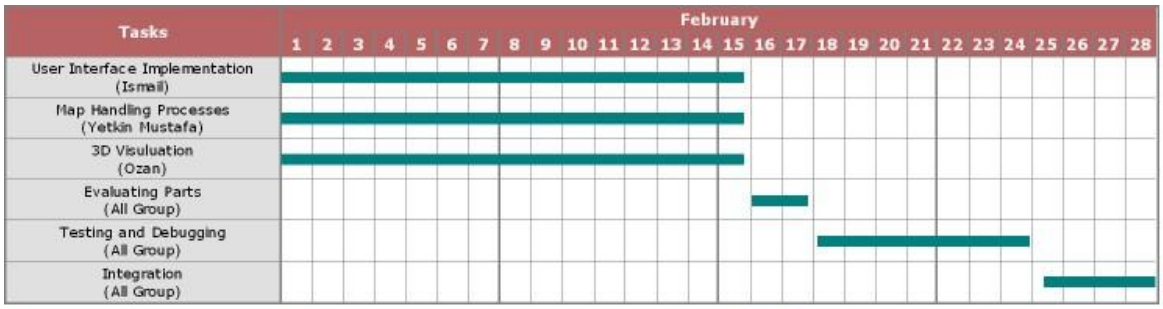
- To illustrate that we can use the current technologies to achieve our final goal
- To have a more specific vision about the final project
- To have a starting point of development of our project

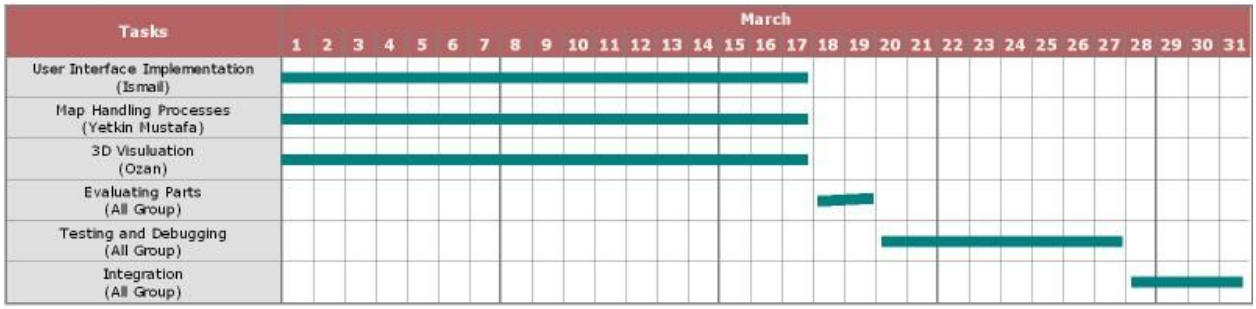
SCHEDULE

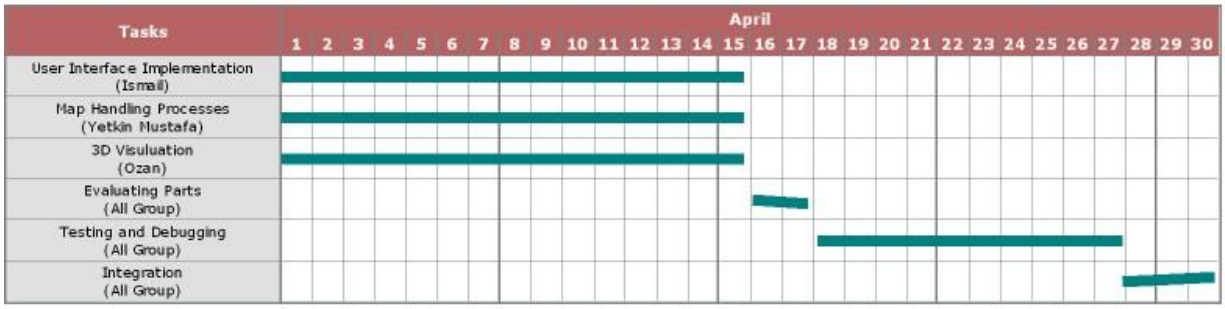


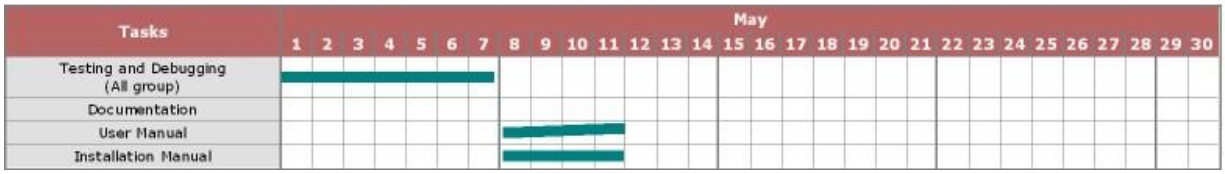

Tasks
November 2008
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
GIS tools research (Ozan)
GIS database research (Yetkin)
Map research (Mustafa)
Mountaineering research (Ismail)
Project Analysis (All group)
Use case modeling (Ismail)
Dataflow diagram (Ozan)
ER diagram (Yetkin)
Requirement Analysis Report (All group)
Architectural Design (All group)
Analysis and requirement review (All group)
User interface design (All group)
Component level design (All group)


Tasks
December 2008
January 2009
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 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Initial design report (All group)
Implementing prototype (All group)
User interface implementation (All group)
Debugging
Prepare for presentation
Final design report (All group)
Prototype demo (All group)
Implementing database
Implementing AI
Testing prototype


Tasks
February
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
User Interface Implementation (Ismail)
Map Handling Processes (Yetkin Mustafa)
3D Visuluation (Ozan)
Evaluating Parts (All Group)
Testing and Debugging (All Group)
Integration (All Group)


Tasks
March
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
User Interface Implementation (Ismail)
Map Handling Processes (Yetkin Mustafa)
3D Visuluation (Ozan)
Evaluating Parts (All Group)
Testing and Debugging (All Group)
Integration (All Group)


Tasks
April
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
User Interface Implementation (Ismail)
Map Handling Processes (Yetkin Mustafa)
3D Visuluation (Ozan)
Evaluating Parts (All Group)
Testing and Debugging (All Group)
Integration (All Group)


Tasks
May
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
Testing and Debugging (All group)
Documentation
User Manual
Installation Manual

CONCLUSİON

We know the importance of well prepared design and we tried to check our initial design in every kind of aspects to make sure it is convenient for development stage and it enlightens the answers to questions about development of Climb Planner software.

We believe that we have corrected some misunderstandings we had led in requirement analysis report.

In this report we tried to make everything as clear as possible. And we believe that we will be using the benefits and outcomes of this report and this report will guide us to develop the Climb Planner software within the following weeks.