

Google Summer of Code Project for KDE

NAME: Marwan Hilmi

EMAIL: mhilmi@gmail.com

IRC: mhilmi on irc.freenode.net

MSN: marwanhilmi@hotmail.com

LOCATION & TIMEZONE: Montreal, Canada. Eastern Standard Time.

CV: http://www.marwanhilmi.com/mhilmi_cv.pdf

PROPOSAL: http://socghop.appspot.com/student_proposal/show/google/gsoc2009/mhilmi/t123878470099

KDE Games: Parsek Client for Thousand Parsec 4X Strategy Game

The Thousand Parsec team began building a multi-platform client which utilizes the KDE API to implement the graphical user interface and the C++ protocol library, libtpproto-cpp, to provide the game logic backbone. Currently, the Parsek client can only connect to servers, download objects, and primitively display messages. The goal of this project is to finish the Parsek client so it can be distributed with KDE Games and allow users to play Thousand Parsec games.

TABLE OF CONTENTS

PROJECT DEFINITION	2
BENEFITTING THE COMMUNITY	3
DELIVERABLES	4
TABLE 1 LIST OF PRIMARY DELIVERABLES	4
TABLE 2 LIST OF SECONDARY DELIVERABLES.....	4
DESIGN STRATEGY AND APPROACH	5
WORK BREAKDOWN AND TIMELINE	6
TABLE 3 PHASE 1	6
TABLE 4 PHASE 2	7
TABLE 5 PHASE 3	7
TABLE 6 PHASE 4	8
TABLE 7 PHASE 5	8
TABLE 8 PHASE 6	9
TABLE 9 PHASE 7	9
PREPARATION STEPS	10
TABLE 10 API AND LIBRARY RESEARCH.....	10
FIGURE 1 REFERENCE MATERIAL	10
FIGURE 2 INTERFACE MOCK-UP	11
AVAILABILITY	12
BIOGRAPHY AND EXPERIENCE	13
TABLE 11 SKILL SET OVERVIEW.....	13
FIGURE 3 PUBLISHED GAME CREDITS	13
CLOSING REMARKS	15

Project Definition

The goal of this project is to finish the Parsek client so that it is in a fully functional state allowing for users to play basic Thousand Parsec games. The idea is to mirror the majority of the functionality provided in the Thousand Parsec 2D Python client, `tpclient-pywx`. The first objective will be to update the existing Parsek codebase to support the latest version of the `libtpproto-cpp` protocol library. Features will then be added to the client to allow for display of the Starmap, reading orders, issuing new orders, displaying the object tree, and reading messages. These features will be implemented by using several components of the KDE API to provide the graphical user interface while the `libtpproto-cpp` protocol library provides the server communication link for retrieving game data. While the KDE API will be extensively used, particular caution will be employed to ensure Parsek remains cross-platform compatible through the use of various available KDE platform ports.

Benefitting the Community

This project's objectives will serve the KDE and Thousand Parsec community in several important ways. The extensive use of the KDE library functionality may will allow for the Parsek client to be distributed through the KDEGames channel, exposing Thousand Parsec to a wider audience and attracting more users to not only play Parsec games but contribute to the project as well. It will also continue the tradition of providing high quality, free, and open-source games to users of KDE. Thousand Parsec acts only as a framework for games and allows users to create and modify their own complete game rulesets. A fully customizable game like Thousand Parsec would prove to be particularly attractive to the typical power tweaking users of KDE.

Since Parsek will be the first full featured client to take advantage of the libtpproto-cpp protocol library, an immediate result of this project will be real world deployment of Thousand Parsec's C++ protocol library. By closely coordinating with the developers behind libtpproto-cpp, several improvements and bug fixes to the protocol library should be expected as a side benefit. In effect, this project will put the libtpproto-cpp library to the test and ensure that it is in a mature state. In addition, the Parsek client will provide viable KDE-based client alternative for users to play Thousand Parsec games. Since this particular implementation will be based on C++, it will be fast and light weight, and allow for users with older machines to be a part of the Thousand Parsec universe without trouble.

Deliverables

Completion of this project will produce several tangible deliverables, ultimately aiming towards having a full feature set in the Parsek client allowing for game playability. The list of deliverables will potentially change as the project progresses either too quickly or too slowly so particular emphasis will be placed on high priority tasks essential for game playability. The number of secondary deliverables ready when the project ends will be dependent on how the project progresses throughout the summer. The bottom line objective is that users should be able to play Thousand Parsec games using the Parsek client. These deliverables and the strategies used to obtain them will be further described in detail later in the proposal.

Table 1 List of Primary Deliverables

ID#	Task	Priority for Playability
1.	Parsek will be updated to supported the latest libtproto-cpp library	HIGH
2.	A 2D Starmap will be displayable in Parsek	HIGH
3.	Functionality for receiving and displaying orders will be added	HIGH
4.	Functionality for creating and issuing orders will be added	HIGH
5.	Visualization of the universe's objects will be added	HIGH
6.	A message display system will be added	MEDIUM

Table 2 List of Secondary Deliverables

ID#	Task	Priority for Playability
7.	Thorough design and code documentation will be provided	MEDIUM
8.	A preference manager will be added allowing users to tweak settings	LOW
9.	A full featured server and account manager will be added	LOW

Design Strategy and Approach

In the undertaking of this project, several factors need to be taken into consideration when choosing an appropriate design approach. Google Summer of Code restricts participants to a rigid timeline of roughly three months so appropriate measures need to be put in place to ensure the project can be completed as required and on time. The design approach will be based on typical waterfall model where we first analyze the requirements and propose a design, and then implement and verify. While there are criticisms to be made of the waterfall method, for this particular short timeframe project, the waterfall method is aptly applicable. Steps will be taken so that the proposed design is evolutionary and can adapt to changes as the project progresses, rather than being a rigid template. Dividing the project into several flexible phases with measurable milestones is also a key step to project success. Ensuring that there is flexibility in the time frames and delivery dates allows for modification of the project should unforeseen problems occur which almost inevitably do. This acts as a simple yet effective form of risk management. Working closely with other developers and mentors will also ensure that when problems arise, they can be quickly handled. Biweekly mentor meetings and weekly progress reports will enforce open lines of communication between myself and developers.

Proper testing and debugging code of any software project is critical so at the end of each project phase with deliverables, builds will be made available to the community so that users can help in bug fixing and also provide feedback. Continual testing and bug fixing will be done through the entire implementation phase of the project so the time required for testing can be minimized in the final stages of the project. I have looked into adopting various graphical interface testing frameworks but the majority of them seem to be commercial applications. Unit testing frameworks like Google's C++ Testing Framework may potentially be adopted to serve testing purposes. I will also setup a local Thousand Parsec server, in addition to using the public demo servers, to test various configurations and data sets. In addition, at the beginning of each project phase I will examine different testing strategies appropriate for each task to ensure that every proposed feature is functional or at the very least, has its potential problems well documented.

Work Breakdown and Timeline

As stated in the design approach consideration, the project will be divided in phases, seven to precise, with distinct objectives, measurable milestones, and deliverables. Work on the project will take place during the regular five day working week, with the weekend time used for emergencies or catch up. The weekdays will be considered working days while the weekends will be buffer time. The buffer time can also be used to implement testing, bug fixing, and working with other developers to resolve issues. Adhering to the timeline as closely as possible is obviously desirable but if unforeseen circumstances arise, flexibility and buffer times are built in.

Table 3 Phase 1: Research and Familiarization

Timeframe	April 20 th to May 15 th 26 days total = 20 working days + 6 buffer days
Objectives	<ul style="list-style-type: none"> • Background reading on KDE4 and Qt4 programming • Setup project blog to monitor progress • Establish regular communication with mentor and setup meeting schedule • Communicate with other project developers • Establish regular communication with mentor and setup meeting schedule • Communicate with other project developers • Setup develop environment and get familiarity with toolset • Refine program design and structure • Create user interface mock-ups
Milestones	<p>Friday, April 24th</p> <ul style="list-style-type: none"> • Project blog is launched and ready <p>Friday, May 1st</p> <ul style="list-style-type: none"> • KDE4 library feature set to be used is documented <p>Friday, May 8th</p> <ul style="list-style-type: none"> • Program structure, design, and feature set is established and documented <p>Friday, May 1st</p> <ul style="list-style-type: none"> • User interface mock-ups are posted for feedback
Deliverables	None at this phase of the project.
Comments	In many ways, this phase of the project is the most crucial. Working with the mentor to establish a solid foundation and direction for the project is an important step to success. The project blog will document progress as the project continues and enable interaction between myself, other mentors, and community members while also providing a platform to receive suggestions and support. The blog will also be used to host reference materials, source codes, and builds so community members have access to the latest materials and tools I will be working with. This will also allow for other developers to pick up where I left off when the project winds down to an end.

Table 4 Phase 2: Library Update

Timeframe	May 16 th to May 31 st 16 days total = 10 working days + 6 buffer days
Objectives	<ul style="list-style-type: none"> • Update current Parsek codebase to work with latest libtpproto-cpp • Make changes in Parsek code to reflect current design and structure • Update Parsek codebase to take advantage of latest KDE API
Milestones	Friday, May 31 st <ul style="list-style-type: none"> • Parsek client is compliable with latest libtpproto-cpp library
Deliverables	Task #1 is completed.
Comments	This phase of the project will involve porting the existing Parsek code to support the latest protocol library. This update will be significant as the newer protocol library allow for asynchronous communication between client and server. Implementing support for this feature will mean the Parsek client will allow for smooth, interruption free gameplay. In doing this, hands on experience working with the protocol library will be gained. I will also be able to identify and resolve any problems or deficiencies in the code on either the client end or protocol end.

Table 5 Phase 3: Starmap

Timeframe	June 1 st to June 15 th 15 days total = 11 working days + 4 buffer days
Objectives	<ul style="list-style-type: none"> • Methods for 2D visualization and drawing using the KDE API will be examined • Update user interface to support widget for Starmap • Implement interface with protocol's GameLayer to receive Starmap data • Draw Starmap data in widget
Milestones	Friday, June 12 th <ul style="list-style-type: none"> • A 2D visualization of the Starmap is implemented in Parsek
Deliverables	Task #2 is completed.
Comments	Finding and implementing an appropriate graphical interface for the Starmap will be the most difficult objective of this phase. This objective will be more complex than simply finding an appropriate KDE/Qt widget as we obviously want to have some unique graphical representation of the Starmap. I believe starting with a simple ASCII based representation of the Starmap and later evolving to supporting 2D graphics would be the ideal approach. I plan on closely working with other team developers and community members to find the proper balance between performance and graphical intensity. In addition, keeping the code modular will allow for the graphical assets to later be replaced or scaled for different platforms.

Table 6 Phase 4: Order System

Timeframe	June 16 th to June 30 th 15 days total = 11 working days + 4 buffer days
Objectives	<ul style="list-style-type: none"> • Implement order interface with protocol's GameLayer • Allow client to receive and read orders • Implement user interface widget to display order queue • Allow client to issue and send orders • Implement user interface widget to display outgoing orders
Milestones	Friday, June 19 th <ul style="list-style-type: none"> • Parsek will have user interface functionality for viewing order lists and queues Friday, June 26 th <ul style="list-style-type: none"> • Parsek will have user interface functionality for sending outgoing orders
Deliverables	Task #3 and Task #4 are completed.
Comments	Implementing both receiving and sending of orders in the Parsek client are two tasks which go hand and hand. The obvious first step will be to implement functionality for displaying other players order and integrate this information with the Starmap and user interface in a usable fashion. The display will be done through a properly formatted textbox widget which focuses on readability. The next step will be to implement an interface for inputting outgoing orders. A general order system widget will encapsulate both the distinct receiving and sending ends and other options common to the order system so they can be easily located and analyzed allowing for maximum usability. The new asynchronous nature of the protocol library will allow for information on either end of the order system to be independently updated as the game progresses. If necessary, some methods of error checking will be performed on outgoing orders to ensure no wasted overhead is spent calling the protocol library.

Table 7 Phase 5: Object Visualization

Timeframe	July 1 st to July 17 th 17 days total = 13 working days + 4 buffer days
Objectives	<ul style="list-style-type: none"> • Implement interface to receive game objects using protocol's GameLayer • Allow client to download and update game objects • Display game objects using a tree or other appropriate visualization • Display each object's unique information in appropriate widget
Milestones	Friday, July 17 th <ul style="list-style-type: none"> • Parsek will have a visualization of the objects in the game's universe and provide the user with the ability to read and manipulate object's unique data
Deliverables	Task #5 is completed.
Comments	Likely using a simple tree widget like Qt's Treeltem would be the appropriate method for displaying the universe's objects but I will investigate other viable alternative which may prove to be favourable. The addition of graphical assets to the tree structure will be a simple way to spice up the user interface but this again will be done in a modular way to allow for easy updating or modification. The design of the interface will again be geared towards readability and usability. Linking the object information to the Starmap will also be an important part of this phase as you would obviously want to highlight selected items on the Starmap. The usage of tooltips will definitely aid in bridging the object tree and Starmap.

Table 8 Phase 6: Message System

Timeframe	July 18 th to July 31 st 14 days total = 10 working days + 4 buffer days
Objectives	<ul style="list-style-type: none"> • Implement interface to download messages using protocol's GameLayer • Implement optimal user interface for displaying and manipulating messages • Allow for real time updating of messages which scroll in textbox widget
Milestones	Friday, June 31 st <ul style="list-style-type: none"> • Parsek users will be able to read, format, and manipulate messages
Deliverables	Task #6 is completed.
Comments	Using the asynchronous nature of the protocol library, implementation of a continually updating and scrolling textbox similar to IRC would be the ideal interface for the message system. Time stamping, logging, and filtering of the messages would also be supported. In the case of text filtering, the KDE API will likely provide an easy method to support this. The widget for displaying messages will be modifiable by the user to allow for options like text color and font size.

Table 9 Phase 7: Final Push

Timeframe	August 1 st to August 17 th 17 days total = 11 working days + 6 buffer days
Objectives	<ul style="list-style-type: none"> • Complete testing and debugging • Ensure functionality critical for gameplay is in place • Provide documentation to future developers for maintainability and expansion • Begin undertaking of secondary objectives if time permits
Milestones	Friday, August 10 th <ul style="list-style-type: none"> • Parsek client is delivered in a state allowing for users to play games Friday, August 17 th <ul style="list-style-type: none"> • Thorough documentation is posted
Deliverables	Task #7 is completed. Task #8 and Task #9 are undertaken.
Comments	The scope of this final phase of the project will be entirely dependent on how the project progresses. At this point, the focus will remain on ensuring the Parsek client is in a ready state allowing for users to play games. Through careful testing and user feedback, if it is determined that Parsek is in a mature state, I will begin working on implementing secondary objectives. In all likelihood, by maintaining proper documentation throughout all the project phases, providing documentation should be a trivial task. The creation of a central preference manager will allow users to modify program settings and tweak the user interface. By initially designing all widgets and user interface elements to be customizable, a preference manager should be an easy feature to implement. The server manager will allow users to save server settings and account settings. Both the client preferences and server settings will be written to plain text configuration files which the user can easily edit or pass to other users. In the case of account settings, a system like KDEWallet may be used to store usernames and passwords.

Preparation Steps

My experience has taught me that early preparation and planning are vital to success. While I am busy juggling end of year projects and exams, I have taken several steps to begin familiarizing myself with the Thousand Parsec project and preparing for the Parsek project. These steps are described below.

- Setting up a GNU/Linux development platform with KDE desktop manager
- Installing the tools and libraries required for this project
- Investigating all related Thousand Parsec documentation available online
- Checking out all related Thousand Parsec software with git
- Compiling different versions of the protocol library, libtpproto-cpp
- Fixing problems related to gcc-4.3 in Parsek so it could compile
- Investigating the existing Parsek code
- Investigating the C++ text-based client to see how it interacts with libtpproto-cpp
- Using other clients to get ideas and inspiration
- Researching the KDE API to see which features can be used
- Creating interface mockups to visualize future objectives and functionality
- Reading reference materials to help prepare

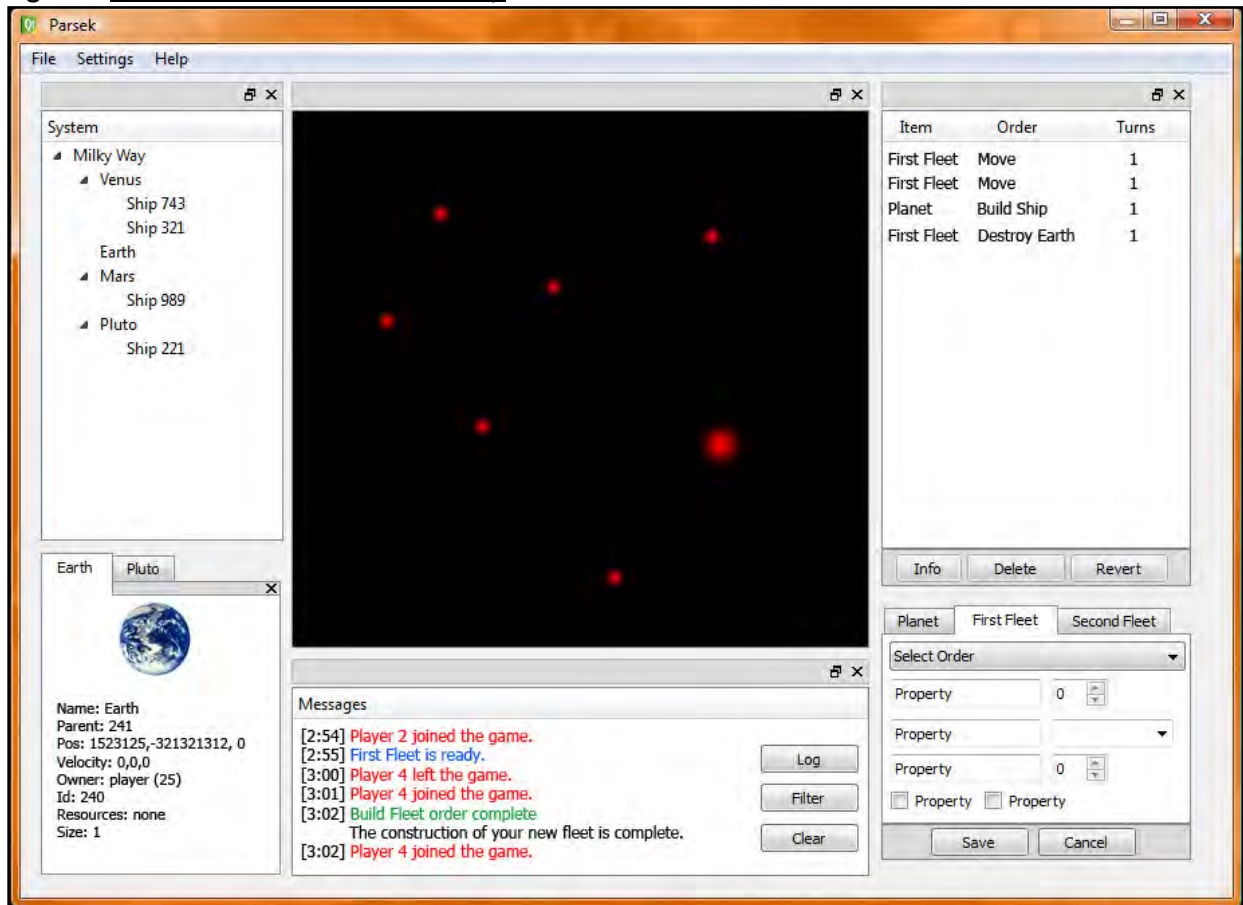
Table 10 KDE API and Library Functionality Researched

Component	Potential Function
KDEUI	To setup graphical user interface and draw widgets
KConfig	For application settings and configurations
KStandardDirs	Helping access resources
KDE Wallet Manager	For storing user passwords
QtTestLib	For testing purposes
Plasma Widgets	For game status and information widget

Figure 1 Reference Material On My Shelf



Figure 2 Potential User Interface Mock-up



In the mock-up shown above, a general idea of the potential user interface for the Parsek client is shown. The distinct systems are all enclosed in dockable widgets that the user can manipulate to their pleasing. The information system below the system tree allows the user to open and close multiple tabs for quick information readouts of different objects. The message system scrolls and will allow for the user to scroll up to view older messages. The order queue is designed to show all the player's outgoing orders for different objects rather than having to select each individual object to view their respective orders. The user can then pull up more information on particular orders or delete and revert orders in the queue. The system to issue orders also uses a tab interface which allows the user to quickly switch between their objects and add orders to the queue.

Availability

My plan is to strictly follow the outlined timeline and work 9 to 5, five day weeks. Actually in my case, my days may be 11 to 7, but the nonetheless, I will be working full time, eight hour days. In the case things do not go according to plan or fall behind schedule, I will use the weekend buffer times described in the timeline to catch up and resolve any issues holding the project back. As stated, I believe by planning the schedule to take into consideration that things may and likely will go wrong, I can effectively manage to keep the project on track. I also hope to carefully coordinate with my mentor to schedule effective biweekly meetings where we can establish a direction for each project phase and coordinate on problem resolution. I plan to be available at all strange hours to accommodate working with mentors in other time zones. I am aware that dealing with developers all over of the world is a crucial part of open source software development.

My last final exam for this school semester takes place on April 24th. I also plan on taking a single summer school course starting May 1st and ending June 15th for approximately two hours every Monday and Wednesday evening. The course load for this class should be easy to manage since it is material I have already covered. My university is basically making me repeat material because of a curriculum reshuffle. It should not prove to be a major obstacle or time hindrance. The class is solely based on the final exam so I will not be occupied with lengthy assignments or lab reports.

I currently work as a part-time developer for a local independent game company. I plan on continuing to work at this position until April 30th at which point I will shift my energy to working full time on Parsek. I have already notified my boss of my plans and he is supportive of my decision and offered his help need be. I also spend a few hours per week training in martial arts and working out at the gym but this is usually on weekends or evenings. Aside from that, I have no planned vacation dates or other obligations and plan on being available for the full duration of the Summer of Code project.

Biography and Experience

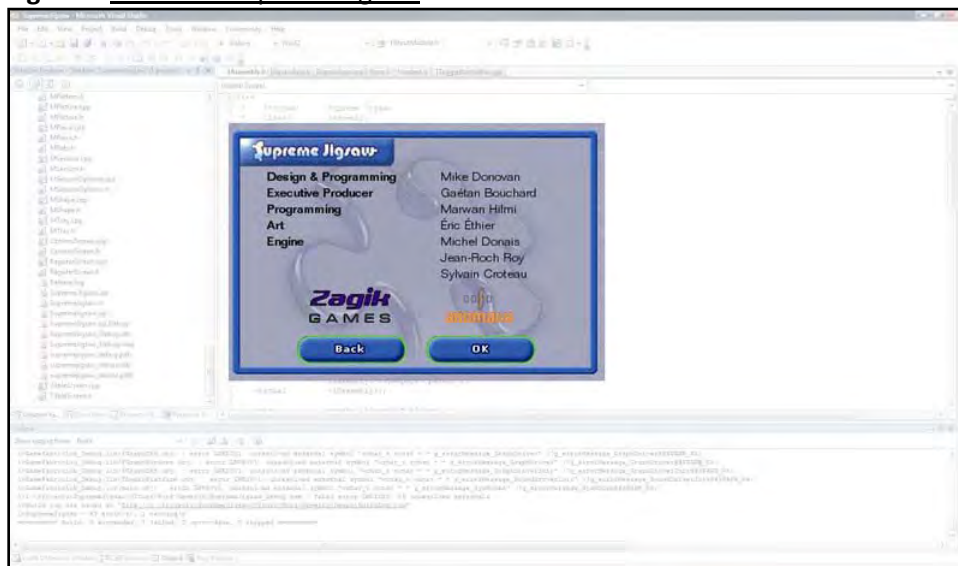
I am a 22 year old student of Computer Engineering at Concordia University in Montreal, Canada. Next fall, I will enter my fourth and final year of my program. My formal education as an engineer has provided me with the right technical and analytic skills required for this project. In practical terms, my education has provided me with skills in C++, C, algorithm and data structure analysis, UML modeling, MC68K assembly, and x86 assembly. The majority of my programming and software engineering courses focused on using C++ to teach modern concepts of program design. I have used C++ for several school projects ranging from simple linked list data structures to complex digital signal processing. My exposure to C and assembly programming comes from my experience with embedded systems and microcontrollers. A course I am currently taking involves building an obstacle finding robot using an ATmega8 microcontroller (only 8K of memory!) programmed in C. I believe my experience with C programming has proven to be extremely valuable at teaching me how to optimize code for efficiency, speed, and memory consumption.

Table 11 Skill Set Overview

Language or Program	Proficiency	Years Experience
C++	Intermediate	4
C	Intermediate	1
Subversion	Intermediate	1
Git	Basic	0
Qt4 Library	Intermediate	1
KDE Library	None	0
Windows and Visual Studio Development Platform	Advanced	4
GNU/Linux and Open Source Development Tools	Basic	0

My experience in the job market has also provided me with significant experience applicable to this project. For the last year, I have been working at a local independent game development company. Our project is multi-platform Jigsaw puzzle game coded in strictly C++. The company's usage of licensed game engine technology has taught me how to integrate with existing code and efficiently use APIs while maintaining fast, modular, and maintainable code. This job has also taught me how to use Subversion for effective revision management. In addition, most of my time at this company has been spent working side by side with a senior developer with over 25 years of experience which has proven to be a fantastic learning experience. A demo of the game is currently available at <http://www.supremejigsaw.com>. Please refer to my CV linked in the front of the proposal for more information.

Figure 3 Credits in Supreme Jigsaw



I participated in Google Summer of Code 2008 working for the ScummVM project. My task was to implement support for audio formats used in the AMIGA version of *The Secret of Monkey Island* and another AMIGA game. Unfortunately the project was a catastrophic failure. I failed to produce functional code which allowed for playback of the audio formats. The project was a failure for many reasons. I spent the summer trying to balance a heavy course load and a part-time job. I did not anticipate the workload required and simply thought it would be an easy task to accomplish. I was never able to get a solid foundation for the project because my planning was poor and inadequate. I also believe my skills at the time were not mature enough to tackle the complex task at hand. Furthermore, I felt my communication with my mentor was poor and I simply did not spend enough time interacting with other developers to receive the proper support I needed. In hindsight however, I was able to learn from many of the mistakes I made and believe that I will not repeat them. I still feel like I letdown the ScummVM team and am disappointed in how the project was unsuccessful which is part of the reason I am dedicated to successfully accomplishing all the goals laid out in this project. Despite the disappointment, GSOC08 was a humbling experience that taught me a lot and working on the ScummVM project introduced me to an extremely large scale project that defies the definition of cross-platform compatible.

My latest hobby includes working on robots and autonomous devices. My friend and I have spent the last couple of years building a large robot weighing several tonnes and capable of moving over twenty kilometres an hour while lifting heavy loads. Most of the programming and interfacing for this robot has been done in C++. My exposure to the Qt toolkit comes from the need to design graphical user interfaces to interact with the robot hardware. In one example, we built circuitry to function as a watchdog shutdown should the robot go haywire and malfunction. The watchdog circuit would ping the onboard computer's serial port and the computer would have to respond within a time interval or the robot would shutdown. Using Qt, I built an application that would connect to the watchdog circuit and display incoming pings and allow the user to manually respond or modify settings. Our latest project involves using Qt to build an interface to allow the user to initialize and control the robot's systems as well as provide real-time readouts.

In my free time, I enjoy reading, programming, practicing martial arts, and playing video games. I have also been a gamer my whole life. I spent a significant amount of my childhood playing *SimCity* on my father's 386 IBM-PC. That machine cost over five thousand dollars and is probably sitting in a dumpster somewhere right now. I later moved onto *SimCity 2000* and *Doom*. My first introduction to 4X strategy games was *Civilization*. I spent quite a few hours playing the different *X-COM* games as well. The game that changed it all for me was *Command and Conquer*. I cannot really say I spent much time playing turn-based strategy games after that but my heart always remained in strategy games.

Closing Remarks

I look forward to working closely with the KDE and Thousand Parsec team to achieve the goals laid out in this project and provide the KDE community with quick access to a wonderful 4X strategy games. This is the only Google Summer of Code task I have applied for and am I dedicated to seeing this project through and taking advantage of the learning experience Google has provided. For the last year, I have wanted to start developing on the GNU/Linux platform and learning more about open-source tools and this project provides the perfect opportunity. The short time I have spent working in the KDE environment has already proven to be exciting. Thank you for your time and consideration.