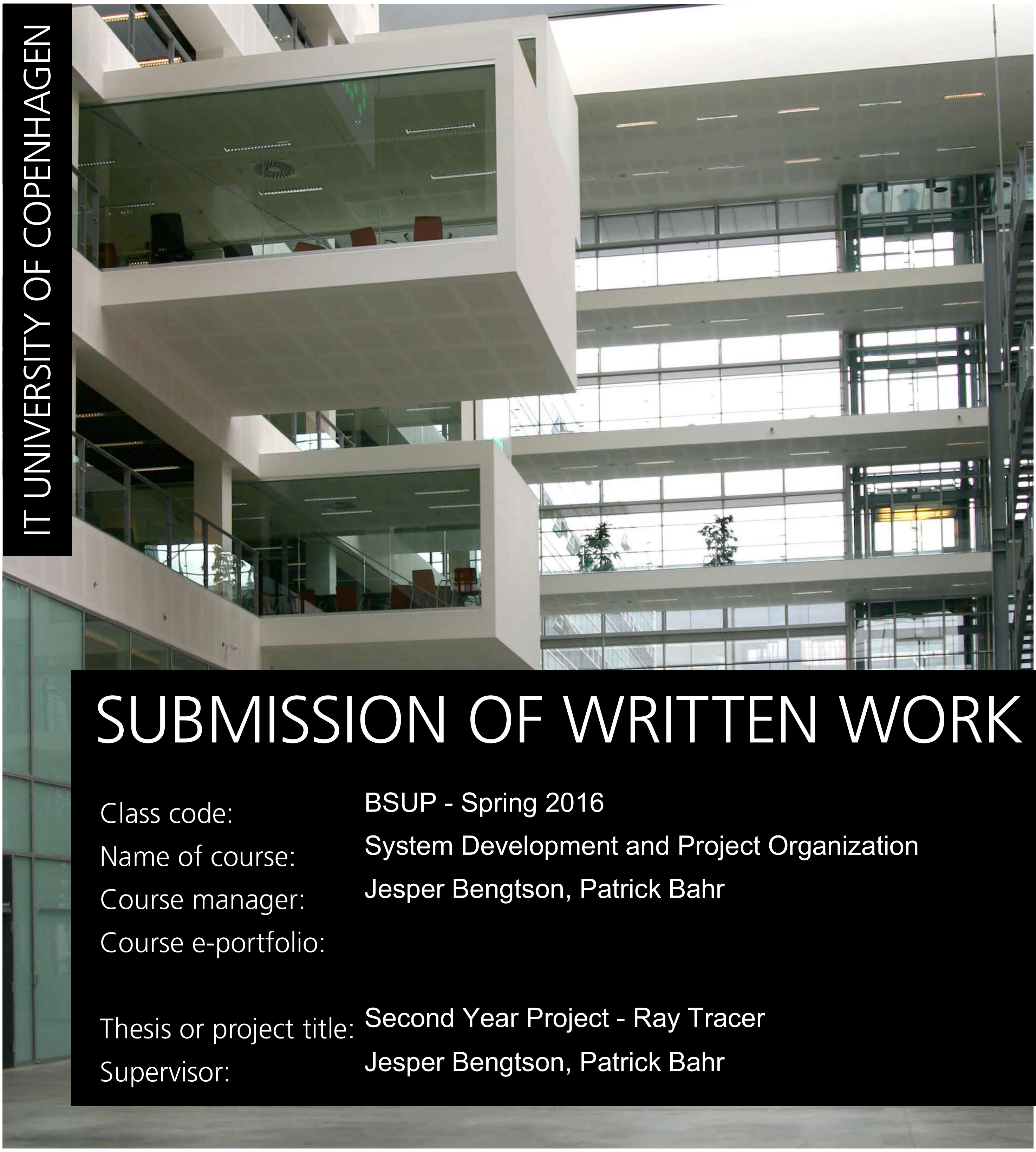


SUBMISSION OF WRITTEN WORK

Class code:	BSUP - Spring 2016
Name of course:	System Development and Project Organization
Course manager:	Jesper Bengtson, Patrick Bahr
Course e-portfolio:	
Thesis or project title:	Second Year Project - Ray Tracer
Supervisor:	Jesper Bengtson, Patrick Bahr

	Full Name:	Birthdate (dd/mm-yyyy):	E-mail:
1.	Daniel Nicklas Rosenberg Hansen	10/04-1994	daro@itu.dk
2.	Dennis Thinh Tan Nguyen	01/04-1993	dttn@itu.dk
3.	Maria Weybøl Techt	12/09-1993	mwey@itu.dk
4.	Nicoline Scheel	22/02-1995	nisch@itu.dk
5.	Emil Refsgaard Middelboe	02/09-1994	erem@itu.dk
6.	Thor Valentin Aakjær Nielsen Olesen	14/02-1995	tvao@itu.dk
7.			@itu.dk

Contents

1 Introduction

The objective of the project discussed in this report is to build a ray-tracer, which can translate three-dimensional objects onto a two-dimensional display, using the functional paradigm of programming through the programming language F#. The implementation of the ray tracer is constrained by a set of functional and non-functional requirements describing the minimal functionality, as well as a strict application programming interface (API) that the implementation must adhere to, and tests that the implementation must pass.

The project was to be developed following the principles of SCRUM, which is an agile method that accommodates changing requirements and circumstances[1]. This makes SCRUM an ideal choice for the project as we were told in advance that the API would be subject to change.

1.1 Initial Viability Analysis

This section describes the initial viability analysis for the project under discussion in regards to the different factors and circumstances that could affect the process.

1.1.1 Requirements

Understanding the requirements and project constraints is important in regards to project planning and task estimation. The theoretical basis of ray tracing focuses largely on vector algebra and light simulation, which the team has limited experience with. With little application knowledge, requirements may seem both ambiguous and esoteric. A large challenge for our project was to understand the requirements initially while the application knowledge was lacking.

1.1.2 Change

These requirements have been presented from the start of the project, but as most requirements, they are subject to change. This introduces another challenge, namely how to plan for change. Since the requirements serve as success criteria of the project, adapting to changing requirements is critical to the success of the project. Agile development frameworks cater for changing requirements through adaptive planning. For our project, we follow the principles of SCRUM, which a minority of the team have previous experience with. Therefore, a considerable challenge is to employ SCRUM correctly or in a way that works for the project at hand and allows for changing requirements. The rapid feedback achieved from SCRUM helps to cope with the derived changes and new requests from the API.

1.1.3 Programming Language F#

As stated above, the system under discussion has to be developed in F#, which the team has little practical experience with. Getting comfortable with a new programming paradigm may lead to delays, especially because the ability to fully benefit from the power of a programming language comes with experience. The fact that the team has to develop new skills requires more time and resources.

1.1.4 Timeframe

Not only is the project constrained by requirements and programming language, it is also constrained by a strict deadline. With limited time available to complete the project comes the question of how to best manage the resources at hand. Conflicting schedules, unavailability, and

[1][Tell, 2016b]

shortfalls of resources are to be expected, and these need to be taken into consideration during risk assessment and planning.

1.1.5 Group Size

Conflicting schedules are not the only difficulty of working within a group of many people. The described project is done in a group of six people. With a rather large group like this, transparency can be hard to achieve, but it is of great importance when it comes to saving resources. Having clear responsibilities enables team members to understand who in the group has an interest in what, and who is responsible for what. Responsibility transparency becomes especially important with a software project such as this, where team members work on different components to be integrated with each other incrementally.

1.2 Initial Risk Assessment

This section describes the initial risk assessment for the project under discussion in regards to both product, process, and resources. Furthermore, it investigates the consequences of the described risks and preventative measures that can be used to reduce the risks.

1.2.1 Product Uncertainty

As described previously, understanding the specification and extent of the product to be developed is challenging. Therefore, it composes a risk towards completing the project successfully. Misunderstanding the requirements may lead to unforeseen work and a discrepancy between what requirements state and what work packages are planned for. That is, not only does understanding the product specification result in rework and time loss, it also introduces a risk of coordination issues. If the project team does not understand the full extent of the project specification, the estimation of tasks will be off, and this might lead to unrealistic or simply bad planning. Furthermore, it complicates pinning out the dependencies between tasks and finding the critical path, which introduces more time loss when team members find themselves dependent on the work of each other. In the project under discussion, it is estimated that product uncertainty will pose one of the biggest risks to the success of the project.

1.2.2 Process Uncertainty

Another group of risks falls under the category of process uncertainty. Choosing a given project plan and making it work can be challenging. The selected plan can be ill-suited for the project at hand, or it can be deployed in a manner that hinders the process. Because of the inexperience with SCRUM within the team, it is a large risk that we fall into the pitfalls of this development process. Some of these pitfalls include excessive planning before initiating sprints - or oppositely, a lack of planning - committing to too many tasks and making assumptions about the process. Luckily, SCRUM caters for a changing process due to the constant feedback from each sprint, which allows the team to adjust the process. Valuable resources may still be lost on process uncertainty.

1.2.3 Resource Uncertainty

Many of the risks inherent to the project under discussion stem from the fact that the project is composed of individuals with different schedules, different backgrounds, and different schemes. For this project group, we have quite varying schedules that conflict at times. This poses a risk that a team member, who is responsible for a particular part of the project, will be unavailable when that particular person is needed. This will inevitably interrupt the development process. Several preventative measures can be taken to reduce this risk, e.g. pair programming and SCRUM reviews. As these tools result in a broader ownership of the project, they lessen the likelihood that there is not a single specialized person available.

As described during the viability analysis, the programming language and application domain is new to the entire project team. The learning curve of F# and the application domain is not considered a large risk since we are all experienced with programming and come from backgrounds introducing vector theory. A larger risk in regards to the competences of the project members is that they differ; some are more comfortable with programming, and some are more comfortable with project management. When collaborating on a project using SCRUM, where all members of the SCRUM team are considered equal[2], the hierarchy of the group may become unbalanced because better argumentation or programming skills may lead to more influence within the team. Team members are at the risk of becoming disengaged if they feel that the equality of the SCRUM team is being overruled. The communication overhead that long discussions introduce can disrupt the progression towards the final product. Preventative measures include establishing clear guidelines as to how consensus is reached when the team is in disagreement and keeping clear communication through daily stand-ups, where every team member get to have their say.

1.3 Belbin Team Roles

We have assembled a diverse team based on the different Belbin-types, as we all acknowledge the importance of having various team roles with specific competences. A Belbin team role is defined as "a tendency to behave, contribute and interrelate with others in a particular way"[3]. Composing a team based on these roles serves for a balanced team, where the strengths of some roles complement for the weaknesses of others. Missing some of the team roles in a team, e.g. in a team with no team players or coordinators, may have a negative effect on the communication. Communication has a tendency to flow easier while having coordinators and team players in a team.

In our team, we cover nearly all the different Belbin team roles. The various roles of each member is outlined below:

Daniel Rosenberg Hansen	Explorer - Driver - Expert
Dennis Thinh Tan Nguyen	Innovator - Explorer - Chairman
Emil Refsgaard Middelboe	Expert - Executive - Driver
Maria Weybøl Techt	Executive - Expert - Analyst
Nicoline Scheel	Analyst - Executive - Innovator
Thor Valentin Olesen	Explorer - Driver - Team player

The only role missing is the completer. The need for this role is, however, not essential as we are all highly motivated and ambitious individuals with a strong desire to reach our goals. Based on the above team role distribution, it is apparent that the team is composed of people who can take a more critical view at things, while others are eager to get the work done. Thus, the weaknesses of one role are balanced out by the strengths of the other roles.

[2][Sutherland, 2013, p. 6]
[3][Tools, 2016]

2 Initial Planning

In this section the initial planning conducted in this project will be discussed in depth. This includes how the plan for the whole project was outlaid, and which considerations were taken into account.

2.1 Identifying the Objectives

When the team got the task of building a ray-tracer, we attempted to establish the project objectives. This was done through the use of user stories describing the functionality of the software. The API for the software would be provided later, and at this time, we could only guess which functionalities would be requirements. Therefore, the initial user stories were imprecise and included many epics.

Upon receiving the API, it was evident that our initial epics matched the requirements well, so we stuck to most of the user stories we had created from the start. The user stories worked as objectives for the project, meaning that an objective driven approach[4] was applied to the project in the initial phase. As the requirements were subject to change, the initial focus was on the core functionality.

Using planning poker, which is described in Section 4.3.1, we created estimated unit values for the user stories. All user stories were inserted along with their estimated values into the product backlog on Trello in order to get an overview of the whole plan. The backlog is discussed further in Section 6. Because of the fact that the requirements of the project were later defined in the API, it proved easier to take the functionality from the API and create tasks from the API itself. In addition, we found that a task is easier to complete than a user story; from a user story one tends to define subtasks, which you have to complete as well in order to implement the given user story. The team decided not to define the product backlog items as user stories to avoid having to redefine the tasks when disassembled into subtasks used for implementation purposes on Trello. This means that new features added to the API were expressed as tasks rather than user stories in the product backlog. Furthermore, the process of building a ray-tracer is quite visual as opposed to other traditional software projects, in which the product progress tend to be invisible[5]. Therefore, you can argue that just from executing the software, you will visually get an idea of the progress of the project.

2.2 Project Characteristics

Many aspects of the project were new to the team. The ray-tracer had to be implemented in F#, which, as stated earlier, is new to the whole group. Everything we knew about design patterns was related to object oriented programming. This means that building a structure for a software in F# did not come naturally to us. Therefore, the risk that the structure could become a problem had already been considered as well as the risk that it would have to be changed at a later time.

During the initial planning, it became clear that many activities in development had many dependencies between each other. In a group of six people, it was hard to find activities that did not depend on each other and did not conflict with the use of resources. To avoid different activities competing for the same resources at the same time, we changed the structural organization and teamed up in pairs. This was done in order to assign more complex activities to each pair and to have a better chance at keeping the resources separated within the team. In addition, this was done to ensure ownership of the code across the pairs, so that a key person,

[4][Bob Hughes and Mall, 2011]

[5][Frederick P. Brooks]

who had implemented a particular part of the system, would always be present. Following this practice helped avoid merge conflicts and duplicate implementations of the same components. Furthermore, when using pair programming, two people are looking at the code simultaneously, which tends to increase the quality of the code more than if written by a single person. This discussion is continued in Section 5.4.

2.3 Practices and Methods

We used the SCRUM ceremonies consisting of daily stand-up, sprint planning, review and retrospective, as well as typical SCRUM artifacts such as a SCRUM board to keep track of our product backlog as well as sprint backlogs. This is described further in Section 3.

It was agreed that short sprints in the beginning of the project would be most beneficial as we would be introduced to new theory frequently within the first weeks that would need to be recorded in the product backlog and implemented. Short sprints meant that we kept the theory fresh in memory. In the first retrospective meeting, it was decided to continue with the short sprint duration that fit well with the overall progress of the project.

As mentioned earlier, Trello was used to maintain our SCRUM board and facilitate the integration of SCRUM in the project. In that way, we were able to access the SCRUM board from everywhere. We created a product backlog which included the product backlog items discussed earlier. In addition, we created a sprint backlog containing a review-, questions- and a done-column. Our Trello practices are described further in Section 4.2.1.

Because of the previously mentioned dependencies, we could have gained a better overview had we created a Gantt chart[6] that clearly stated specific goals that would need to be reached at some point. Also, showing which activities depended on one another by making an activity network would have paid off in regards to planning.

We did not do these things, partly as a result of the domain logic being too ambiguous to get an overview of which activities depended on which, before having deep insight into ray tracing theory. This means that creating a Gantt chart and finding the critical path were not a part of our initial planning of the project.

This could have improved not only our initial planning stage, but future planning as well, since it would have let us prioritize activities better. This would have meant we would not get stuck with something depending on another component to be implemented first and thus prevented idle time. Having a Gantt chart would have given us a much better overview of the project as a whole. The goals would have been much clearer, and we would not have had such a struggle defining milestones.

Instead, we used SCRUM extensively, and in the sprint planning of every sprint, we discussed what was most important. This was a time consuming process in the beginning, mainly because of the ambiguous domain logic, and the fact that we had no grasp of the bigger picture of the application domain. As time passed, we gained a better overview of the program and the sprint planning meetings got shorter as our knowledge of the application domain expanded.

[6][Bob Hughes and Mall, 2011, p. 65]

3 SCRUM

This section describes some of the motivation behind following the agile SCRUM process in this project. In addition, the section looks into some of the benefits and limitations of using SCRUM in a project as opposed to other traditional software process models like the Waterfall model.[7] Finally, the experience gained by the team using SCRUM during the project is encompassed and used to describe our view on SCRUM.

3.1 Motivation

The SCRUM framework was integrated into the project being a mandatory requirement specified in the System Development and Project Organization course. However, many other reasons may be used to justify the need and use of SCRUM in this particular software project.

3.1.1 Minimum Viable Product

Scrum is known as an agile process that allows one to focus on delivering the highest business value in the shortest time.[8] The software process model strives to increase productivity and remove impediments by delivering products through an iterative and incremental process with open feedback. One of the benefits of the incremental product deliveries is based on the fact that a version of a working product is always available. Taking point of departure in the project, one of the first sprint goals we set out to complete was to demo a sphere on the screen when running the ray tracer program. This incremental release was our first MPV (minimum viable product). Through the use of SCRUM, we created a product with the lowest set of product features that could give value to users, both to test the concept of ray tracing and get feedback before continuing development. As a result, we were able to get an idea of the core principles inherent within the domain of ray tracing while also avoiding building a product that does not comply with the requirements outlaid by the client or end-users. However, it should be noted that we do not have any real users of the product. The product is solely based on a set of predefined functional requirements and feedback given by our professors in the project. Thus the product was never supposed to be released but is instead used for educational purposes.

The outcome of SCRUM is analogous to the famous "Lean Principle"[9] used when running a startup. The lean software development process focuses on a 'build-measure-learn feedback loop' comprised of validated learning to achieve the same minimum viable product thus maximizing customer value while minimizing waste.[10] This is similar to the incremental 'Done' deliveries in SCRUM projects that ensure a version of the working product is always available.

3.1.2 Communication and Transparency

One of the main motivations and benefits of using SCRUM in the project has been the improved communication. The process has promoted the communication, cooperation and collaboration within the team dramatically. As a result, the project has been exposed to fewer misunderstandings as opposed to previous projects that contained less clarity about the domain and workload within the team. The daily scrum meeting was especially helpful to give an overview of each others work. On each day of the sprint, our team held the meeting going through the three questions of; what we did yesterday, what we were going to do on the day and finally whether anything was in our way.

[7][Wikipedia, 2016]
[8][Tell, 2016b]
[9][leanmanufacturingtools, 2016]
[10]http://theleanstartup.com/principles seen 16h of May, 2016.

These answers served as commitments in front of our fellow peers and helped 'synchronize' the group work. Furthermore, the last question contributed to removing any impediments in our way and increase the productivity of each team member leaving us with less idle time where less work would be done. The clarity and overview made it easier for us to adapt to changes quickly and confront any possible misunderstandings that might occur in the work process, product changes or understanding of the domain. In addition, it should be noted that to achieve this transparency it helped a great deal having a SCRUM master to facilitate communication, keep track of things and coordinate the daily SCRUM.

3.1.3 Domain Uncertainty

Arguably, the SCRUM process was a good match for the project due to the previously mentioned occurrence of changes. However, it was also ideal to cope with an unknown domain that none of the team members had any prior experience with. The holistic approach gained from SCRUM along with the iterative approach helped us narrow down the scope of the project, acquire knowledge on the domain and adapt to changes.

Initially, we were not able to clearly define the extent of the project and were thus not able to make precise estimates. In this regard, SCRUM was ideal to use because none of us has worked on similar projects before leaving us without any valuable experience that could have been used for precise estimation on tasks. Arguably, one could have considered other alternatives like the waterfall model if we had already done similar projects and thus knew how long it would take, leaving us with better planning and estimates. However, due to the uncertainty of the project and the domain, we did not even consider the waterfall model. In general, the waterfall model does not allow the team to react quickly to changing business conditions which frequently occurs in today's global world of business.

3.2 Software Process Configuration

This section describes how we configured and integrated the SCRUM framework into the project. Specifically, it describes how the practices were implemented along with the team member roles and responsibilities within the team.

We strove to use the roles, events and artifacts examined throughout the lectures of the course. Firstly, this meant consequently going through the four activities of sprint planning, daily scrum, sprint review and sprint retrospective for each sprint.[11] Secondly, we made sure to delegate the product owner and scrum master roles within the development team.[12] Thirdly, we worked on the initial user stories and created a list of all features based on the functional requirements outlined in the Second Year Project.[13] These requirements were captured as items to be used in the product backlog.[14] The product and sprint backlog were made available for the whole team using the collaboration tool "Trello" that organized the project into boards. The electronic board gave the same overview of what is being worked on, who is working on what and where we were in the process similar to the purposes of a physical scrum board.

3.2.1 Activities

We followed the four activities used in SCRUM sequentially for each sprint.[15] The duration of the sprint was set out to be one week long initially which we deemed fit due to the changing

[11] [Point, 2016d]
[12] [Point, 2016b]
[13] [Point, 2016c]
[14] [Point, 2016a]
[15] [Point, 2016d]

nature of the project and API. In the sprint planning, we would produce a sprint backlog based on items from the product backlog and set out a sprint goal. This might have been something to demo in the sprint review where the team would present what is accomplished during the sprint. By way of example, we set out to a demo showing a sphere on the screen in the first sprint and later a triangle in one of the other sprints. The sprint retrospective was used to inspect and adapt the SCRUM process while also discussing the things that went well and things that went poorly. This allowed us to improve the sprints consistently for our needs and helped continuously optimize the working environment to increase productivity. Finally, the daily scrum was held on each sprint day to give an overview for all members of what others were working on and thus "synchronize" the group work.

3.3 SCRUM Limitations

This section describes some of the limitations we experienced using SCRUM and what experience we gained from it. However, we did not follow the practices strictly: firstly, we did not have an actual product owner even though we chose a team member to take up the role in the start of the project. As a result, the scrum master tended to act like the product owner which caused some misunderstandings and conflicts within the group and the sprint planning. Secondly, we did not always stringently do the sprint review.

Also, we were not able to set out any extensive initial plan and project goals, which we did not feel SCRUM supported in the same way as a more sequential approach like in the waterfall model. Finally, the SCRUM framework requires much effort from a development team and may kill productivity if not managed appropriately and within certain time limits. However, this is most likely the case for all kinds of process methodologies and the problem did diminish as we gained experience with SCRUM and became more familiar with the activities. In this matter, it will likely help a great deal being introduced to the framework now so that we are familiar with it and may use it in future projects.

4 Technology Ecosystem

We have adopted several technologies to support different aspects of the project. This section will provide a description of the tools, why they were chosen and what influence they had.

4.1 Communication

4.1.1 Meetings

Today's technology makes it a lot easier to communicate across distances - however, face to face communication is always to be preferred. There are numerous advantages - misunderstandings are easily avoided, and it is a lot easier to explain a problem and get solutions from other participants. Of particular technologies, we used the daily scrum, which helped us gain an insight into work that had been done, what problems had arisen, and speak of our hindering with the project.

4.1.2 Facebook

We utilized Facebook as a means to communicate whenever the group was unable to meet. It was deemed to be the most suitable means of virtual communication due to it being one of the most commonly used communication platforms today. Hence, a message is rarely overseen and can be instantly reacted upon when received. As opposed to the Facebook instant messaging, emails, for instance, may have slowed the communication process significantly as one might not check their email often. Emails would only have been appropriate if we had great temporal distance between the team members and one had to consider different variables such as time differences.

4.1.3 Slack

The communication tool Slack appeared to be an excellent option for the group to use as a communication platform. We decided to try it out based on the fact that it omits a lot of the advertisements and distractions experienced while communicating via Facebook. After a short time, the tool was deemed unnecessary for the group as Facebook fulfilled the need for virtual communication and did not conflict with our goals. This may partly be the result of Slack having more utility, but being less integrated into the group members' daily lives. As a result, Slack was more cumbersome to use as the main virtual communication tool. As such, shifting to Slack would require each member to change their habits regarding which communication platform they use. Consequently, the group deemed this to be wasteful and preferred to spend energy on something more relevant to the project.

4.2 Organization

4.2.1 Trello

We needed an efficient way to get an overview of our Scrum board. Ideally, we should have a row for the product backlog, one for the sprint backlog and one for the tasks done. Trello allowed us to do just that. It was later decided to separate the finished tasks into boards belonging to their respective sprints so that we knew which sprint backlog items had been completed in what sprints.

4.2.2 Google Drive

Google Drive have proven very useful for keeping shared notes on our SCRUM processes, such as the daily stand up, retrospective and scrum planning. The tool is readily available, easy to use and makes coordination of shared notes between group members a nonexistent problem. Google Drive has also been utilized in the process of collective brainstorms in which all team members could participate and contribute to the same brainstorm document at the same time.

4.2.3 Overleaf

Given that the group agreed upon creating the report using latex, there was a choice to be made. Either one could use an online service such as Overleaf to coordinate the writing process of the report or use version control in conjunction with local latex compilers. The group decided to use Overleaf as the online tool to coordinate the writing of the report as several team members already had good experience with the service, and it was quite straight forward. Also, the updates are shown fast and does not require tedious commits - in this regard, it works like Google Drive.

4.3 Coordination

4.3.1 Poker Planning

We used poker planning as a means to estimate the capacity of tasks located in the sprint backlog. Poker planning is a way of putting units on each task where each team member make an educated guess as to the estimation, and then everybody in the group reveals their guesses. Through argumentation, the most likely estimation is settled upon.[16] This proved rather helpful as a way of knowing how much effort and resources were required for a given task and thus allocate them accordingly. Taken that into consideration, poker planning also allowed the group to estimate better how many tasks could realistically be completed within each sprint. Finally, the estimations and units were used to illustrate the progress of the project in burn down charts and thus provide a clear status of the project.

4.4 Awareness

4.4.1 GitHub

GitHub was used as our version control platform. We used an approach with inspiration from what is called 'Git flow'[17]. In 'Git flow', branches are used every time a new feature is to be worked on and thus prevent errors from occurring when working concurrently on files shared between members, while also allowing one to revert any broken work. The group utilized a two-main-branch structure. A branch called 'Developer branch' is used for sharing latest work, which may not be ready for release yet. Finally there is a master branch, in which weekly releases of stable working features are shared.

The reason for not using the actual git flow is simply based on the fact that we did not actually need to support continuous stable releases. The git-flow technique was mainly used to maintain a healthy GitHub structure in order to minimize conflicting work and continuously support an incrementally working release. Descriptive comments were used so that we always knew exactly what had been done in a certain commit. However, there were times where code

[16][Tell, 2016a]
[17]atlassian [2016]

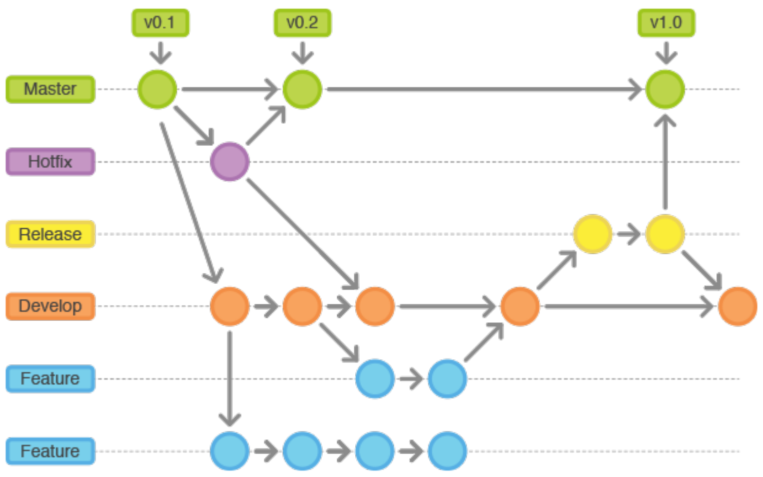


Figure 1: Model of git flow approach.

disappeared because of Github, but these occurrences were minor and only required us to merge changes in the product manually a few times.

Further, we decided on a set of good practices from the start that we would have to follow throughout the project: firstly the master branch should always be kept “stable” in the sense that it should be able to compile and run. The only time we pushed into the branch would be at the end of a sprint. The development branch was to be kept “stable” as well - however, it was permitted to push to this branch at any time. All functionality should therefore be implemented on separate branches. For these branches, the naming convention seen in Table 1 was used.

feature/*	This was used for new features
restructure/*	This was used for refactoring
fix/*	This was used for bugfixes

Table 1: GitHub naming convention

The asterisk (*) represents the title of the current activity.

We decided to merge all currently working functionality into the master branch after each sprint and make an incremental pre-release based on this. The pre-release would include all the functionality from the sprint backlog for which the sprint would be considered successful.

5 Cooperation Practices

The following section focuses on the practices we have chosen to use in extension to the main SCRUM practices described in Section 3.

5.1 Subversions

After a finished sprint, we made it a practice to merge everything that worked into the master branch of our GitHub. We then created a pre-release for this new functionality. This worked for us as a sub-goal that the team had reached.

The goal was always to implement all the functionality in the sprint backlog. However, the goals were not always reached and were therefore postponed to the next sprint.

We had a specific naming convention for the pre-releases by using animal names starting from A. We sometimes tried to find an animal that symbolized the progress of the sprint. For instance, the Dodo release was a release with minimal new functionality but filled with refactoring and bug fixes.

5.2 Team Organizational Structure

We strove to have a flat and democratic team structure. In our team, we had a SCRUM master and some people who had acquired more knowledge of the domain than other team members. The team wanted to be able to discuss freely among each other; no titles or prior knowledge should make one's opinion more valid than others'. Our SCRUM master was thus providing the administrative leadership at our meetings only, as nobody felt the need to go directly to the SCRUM master with any problems. Instead, issues within the team were brought up during meetings.

Also, the team strived for egoless programming[18] where team members were allowed to review one another's code without offending them. In this regard, we all had to be on the same level. In addition, it is shown that members of a SCRUM team will have a higher moral and job satisfaction if they have a saying in the decision making progress. Further, in less understood projects, a group of programmers is faster at coming up with a solution than an individual[19]. Our team should have a common goal, and all voices should be heard and respected, which was encouraged through team building.

5.3 Team Building

To strengthen our teamwork, we arranged many informal meetings where the team members could socialize with each other more loosely. Communicating with each other informally were part of building trust to one another. This made it easier to ask for assistance in work situations as it flattened out the organizational structure within the team. The fact that we got to know each other more informally also changed the jargon in the team, making working together more enjoyable.

5.4 Pair Programming

During the sprints, we mainly programmed in pairs. The discussion about solving the different problems is often more helpful than dwelling on a problem by oneself, which can make the process quicker - two minds think better than one. However, in some cases you may argue that

[18][Bob Hughes and Mall, 2011, p. 270]
[19][Bob Hughes and Mall, 2011, p. 278]

the development effort is doubled when working together in this manner. But if one were instead to do peer reviews of code that has been written individually, the person reviewing would have to study the documentation before being able to make the review. This can also be a very time consuming progress.

When working in pairs, we found that the attention you get from someone who is staring at your code and pointing out every mistake you make can make you feel pressured. It did, however, minimize the amount of hitches that one may suffer when not inspecting each others work.

We found that this practice worked well for this project as it was filled with dependencies that made it hard to distribute six different tasks among the team that did not depend strongly on one another. Instead, we were split into teams where we could work on more independent tasks and avoid blocking each other from being productive throughout the sprint work.

The division we created in the first sprint became the division throughout the whole project, in contrast to creating new teams for every sprint. This is not normally considered good practice[20], but we did this because of our short sprints and because we found that the pairs worked well together. So instead of trying to mix things up, we stuck to what we knew worked.

We were able to help each other across the teams, though: if one pair was stuck in a problem, a member from another pair could join them in order to find a solution.

5.5 Explicit Communication

When talking about communication, one often divides it into implicit and explicit communication [21]. Explicit communication is the kind of communication that one can follow without having a larger knowledge foundation, as opposed to implicit communication that is based on "implied" talk where one needs a knowledge foundation to understand what is being said. We had a couple of practices in our teams to strengthen our explicit communication - in the breaks we casually talked about the program, which is a practice that we had the opportunity to use because we were located on the same geographical location.

The team repeated this at the informal meetings where the communication were generally characterized as being more casual, spontaneous and personal. Also, on many occasions, we walked the whole team through some of the more crucial parts of the system. This was done to see if the implementation was readable, and to inform the peers of the structure of the system and thereby develop a common knowledge foundation and common naming conventions.

When a team member got an idea for implementation, we strongly encouraged the team member to draw it while explaining the idea. This helped the peers understand the idea and minimize any possible misconceptions. Furthermore, we used analogies whenever possible to make it easier for people to understand. These are practices that we have found to work very well in the process of utilizing explicit communication used to solve problems in the project.

[20][Bob Hughes and Mall, 2011, p. 91]
[21][John Noll and Richardson, 2010]

6 Initial Backlog

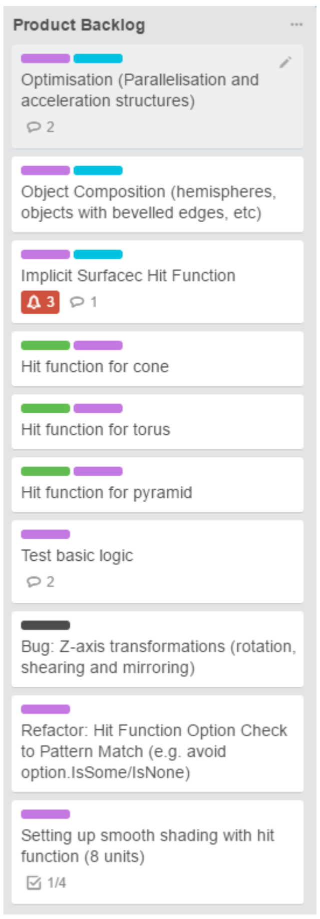
Our initial backlog was developed before we had come to a full understanding of the project. Thus, it only provided an overview of the requirements as briefly stated by the client. We mainly focused on the needs that an outside user of the program would have and did not go into detail with the programming. Furthermore, we created as many epics as we thought needed, which we intended to elaborate on in much more detail when the requirements became clearer and we gained insight into the theory needed to develop the ray tracer.

6.1 Backlog

- As a designer, I want to simulate ray tracing so that I can make realistic and beautiful pictures
- As a designer, I want to be able to simulate the way in which light, shadows and reflection behaves around objects
- As a user, I need objects represented in three dimensions
- As a designer, I want to be able to load a scene into the program and display it
- As a user, I want to render the Stanford Bunny without experiencing a delay above 20 seconds
- As a user, I want to manipulate an object's appearance
  - I want to manipulate the shape
  - I want to manipulate the scaling
  - I want to manipulate the rotation of objects.
- As a designer, I want to manipulate an object's material
  - I want to manipulate the texture
    - I want to assign a texture to an object
    - I want to change the texture of an object
    - I want to be able to assign color as a texture for an object
    - I expect the object to have a standard texture when inserted
  - I want to manipulate the gloss
    - I want to change the gloss of an object
    - I expect the object to have a standard gloss when inserted
  - I want to change the transparency
- As a user, I want objects to be rendered efficiently so that I don't have to experience delay
- As a user, I want to be able to change the amount of details in an image
- As a user, I want to see reflections of my objects realistically drawn, taking lights into account

6.2 Late Backlog versus Initial Backlog

Obviously, the later version of the product backlog does not contain the same amount of missing work. This is simply based on the fact that PBI's have been completed throughout the process. Interestingly, the product backlog items (PBI's) in the later backlog look quite a bit different from the first one, as illustrated in the figure beneath. This is the product backlog taken from Trello as it looked during sprint 'Fox'.


Product Backlog

Optimisation (Parallelisation and acceleration structures)
2

Object Composition (hemispheres, objects with bevelled edges, etc)

Implicit Surfacec Hit Function
3 1

Hit function for cone

Hit function for torus

Hit function for pyramid

Test basic logic
2

Bug: Z-axis transformations (rotation, shearing and mirroring)

Refactor: Hit Function Option Check to Pattern Match (e.g. avoid option.IsSome/IsNone)

Setting up smooth shading with hit function (8 units)
1/4

Figure 2: An example of a late product backlog

There is quite a bit of difference between typical user stories and the tasks used to define work items internally within the team. Whereas the tasks describe actual coding work to be done, user stories are usually defined by a user's specific needs as well as a number of conditional criteria for these needs to be implemented. Since most of the requirements were given beforehand, the payout for creating user stories in order to derive requirements was not very big compared to the amount of time and resources it would take from the group.

The group learned throughout the process of the sprints that a lot of the PBI's from the initial backlog was quite ambiguous and needed to be separated into smaller and more specific tasks. The problem with broad or ambiguous backlog items is that they are hard to evaluate the work volume of. Determining the subtasks of the given PBI gives a much better understanding of the task as a whole and makes it easier to do the velocity calculation ultimately used to make a burndown chart that helps predict when all of the work will be completed.

7 Sprints

The following section presents and discusses how the sprints were organized, performed and reviewed in the project. The progress has been measured using burn down charts to clarify the amount of work left after each sprint. This has ultimately been used to see if the project was on track and would meet the functional requirements of the product before the deadline.

7.1 Sprint Setup

Each sprint was defined with a fixed duration of one week. This was decided based on the initial amount of work and estimated work hours that fit well with one-week sprints. Also, the API would be subject to weekly changes and the decided sprint time span would cope effectively with this.

We decided on two weekly sprint days on Tuesdays and Thursdays where the team would meet. The first workday began on Tuesdays where planning, reviewing and a retrospective was made whereas each sprint ended on Mondays. This decision was based on the availability of teacher assistants for the project, but also the available time of each member.

The outcome of each sprint was a potential release that was titled with an animal name starting from A for easy reference of version. After seven sprints, the following potential releases have been delivered:

- Armadillo
- Baboon
- Cuttlefish
- Dodo
- Fox
- Giraffe

7.2 Sprint Planning

The sprint planning was set to allocate in between one and two hours each Tuesday where the product backlog was discussed. The discussions were based on what had previously been done and if there was something undone from the previous sprint. Undone items were either included into the next sprint or broken down further. Unfinished sprint backlog items mainly occurred due to underestimation or unforeseen work dependencies within the group. By way of example, the implementation of the data structure used to store objects in the ray tracer depended on some of the completed content from previous sprints. It was only estimated to take one sprint, but ended up taking three. Thus, this feature was broken further into small sub-parts and added to each sprint until completion.

7.2.1 Velocity Ranges and Prediction

This subsection will discuss the notion of velocity ranges and how the group made their prediction. Items and predictions were based on previously completed items and units derived from poker planning. After six sprints, a velocity range of the team was calculated. The velocity range is an agile project management tool that is utilized to predict the amount of work a team will complete after each sprint, but also a planned set of future iterations.[22] Based on the number of the completed units of the team (29,40,41,48,54,97) the median velocity has been calculated to be 44.5 units per sprint along with a 90% probability of the velocity being in between 29 and 91. 29 is the minimum amount of work the team will complete while 91 is the maximum amount of work the team may complete. This does not mean one will produce these amount of work every time as unexpected situations may occur and therefore give a different velocity at the end of a given sprint. Hence, a velocity is a range and not a fixed number.

The calculated velocity range allowed the group to give an approximate prediction on how many units the group would be able to deliver for each sprint. However, it should be noted that it would require a certain amount of sprints for the numbers to be relatively precise. After a few sprints we had enough data from our log of estimations and burndown charts to somewhat safely forecast our future sprint deliveries. Initially, the team and its members did not have any knowledge or experience within the domain and the derived estimates were rather inaccurate. Gradually, these estimations have been improved and we have gotten a better understanding of how much value we could deliver during sprints. This gave us the ability to assume that we could deliver certain items in the product backlog with a certain confidence, e.g. within 90% confidence range we would be able to deliver item x at time y. Altogether, these estimations and burn down charts should ideally reach 0, meaning we have delivered all forecast items in the product backlog within the final deadline.

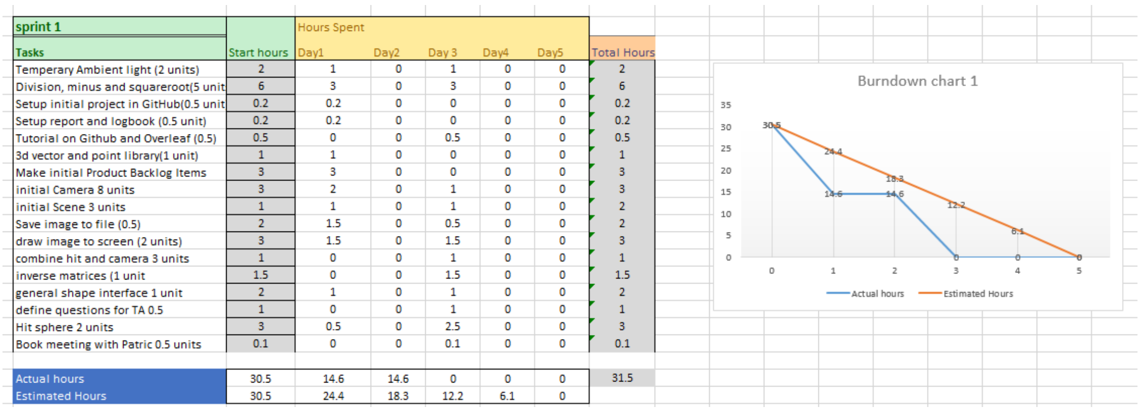
7.2.2 Burndown Charts

The following subsections go through the sprint burndown charts created during the project and aims to enlighten some of the numbers and how they were derived based on the respective tasks outlined in each sprint. Burn down charts is a graphical representation of remaining work versus a fixed time frame. The chart is useful in the sense that it allows one to predict when all work will be completed as well to track the progress of the project. This can be seen by how the actual burn down line is aligned to the estimated burn down line. If the actual burn down line is above, one is behind schedule according to that given point in time. If the actual burn down line is below, one is ahead of schedule.[23]

[22][Software, 2016]
[23][Dinwiddie, 2009]

7.2.3 Sprint 1

At the initial phase of the project, no member had any required knowledge of the domain and thus no precise estimate of expected time and resource to be made on the initial tasks.



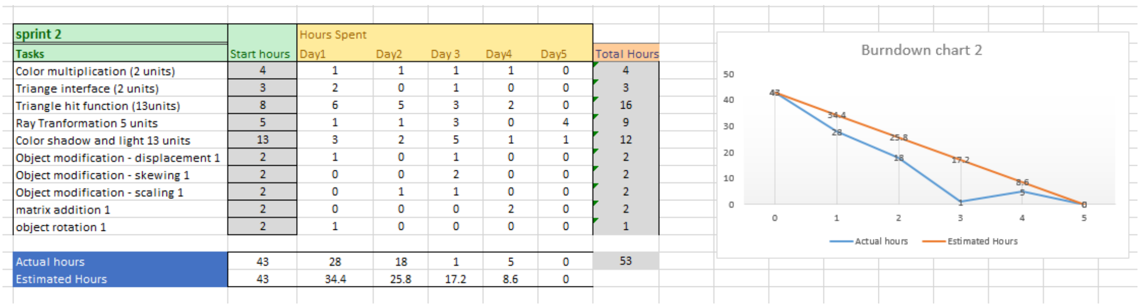
sprint 1		Hours Spent					
Tasks	Start hours	Day1	Day2	Day 3	Day4	Day5	Total Hours
Temperary Ambient light (2 units)	2	1	0	1	0	0	2
Division, minus and squareroot(5 unit	6	3	0	3	0	0	6
Setup initial project in GitHub(0.5 unit	0.2	0.2	0	0	0	0	0.2
Setup report and logbook (0.5 unit)	0.2	0.2	0	0	0	0	0.2
Tutorial on Github and Overleaf (0.5)	0.5	0	0	0.5	0	0	0.5
3d vector and point library(1 unit)	1	1	0	0	0	0	1
Make initial Product Backlog Items	3	3	0	0	0	0	3
initial Camera 8 units	3	2	0	1	0	0	3
initial Scene 3 units	1	1	0	1	0	0	2
Save image to file (0.5)	2	1.5	0	0.5	0	0	2
draw image to screen (2 units)	3	1.5	0	1.5	0	0	3
combine hit and camera 3 units	1	0	0	1	0	0	1
inverse matrices (1 unit	1.5	0	0	1.5	0	0	1.5
general shape interface 1 unit	2	1	0	1	0	0	2
define questions for TA 0.5	1	0	0	1	0	0	1
Hit sphere 2 units	3	0.5	0	2.5	0	0	3
Book meeting with Patric 0.5 units	0.1	0	0	0.1	0	0	0.1
Actual hours	30.5	14.6	14.6	0	0	0	31.5
Estimated Hours	30.5	24.4	18.3	12.2	6.1	0	

Figure 3: Burndown chart of Sprint 1

Initially, it may be seen on the first burn down chart (see figure 3) that each item was seemingly over-estimated. Hence, the group was done with all items already on day 3. Most of the items added were based on work related to starting up the project, booking meetings and creating the initial structure which did not require a full sprint.

7.2.4 Sprint 2

The second sprint was more consistent with the estimated time of 43 hours of items. The team was almost done on day three, but the group stumbled upon implementation problems that required additional resources and time (see figure 4). Ultimately, this sprint moved us towards a better prediction on how many units we could complete on a sprint.

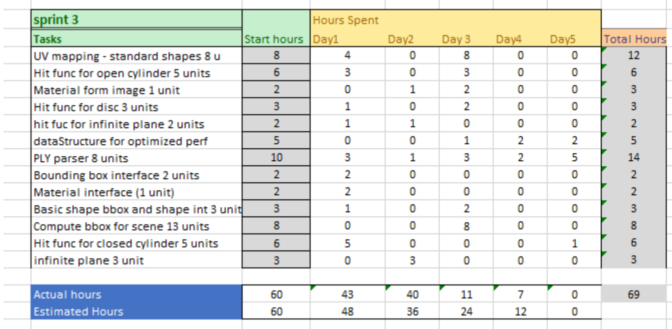


sprint 2		Hours Spent					
Tasks	Start hours	Day1	Day2	Day 3	Day4	Day5	Total Hours
Color multiplication (2 units)	4	1	1	1	1	0	4
Triange interface (2 units)	3	2	0	1	0	0	3
Triangle hit function (13units)	8	6	5	3	2	0	16
Ray Tranformation 5 units	5	1	1	3	0	4	9
Color shadow and light 13 units	13	3	2	5	1	1	12
Object modification - displacement 1	2	1	0	1	0	0	2
Object modification - skewing 1	2	0	0	2	0	0	2
Object modification - scaling 1	2	0	1	1	0	0	2
matrix addition 1	2	0	0	0	2	0	2
object rotation 1	2	1	0	0	0	0	1
Actual hours	43	28	18	1	5	0	53
Estimated Hours	43	34.4	25.8	17.2	8.6	0	

Figure 4: Burndown chart of Sprint 2

7.2.5 Sprint 3

The third sprint consisted mainly of work related to the core functionality. The actual hours versus the estimated hours were consistent (see figure 5).



sprint 3		Hours Spent					
Tasks	Start hours	Day1	Day2	Day 3	Day4	Day5	Total Hours
UV mapping - standard shapes 8 u	8	4	0	8	0	0	12
Hit func for open cylinder 5 units	6	3	0	3	0	0	6
Material form image 1 unit	2	0	1	2	0	0	3
Hit func for disc 3 units	3	1	0	2	0	0	3
hit fuc for infinite plane 2 units	2	1	1	0	0	0	2
dataStructure for optimized perf	5	0	0	1	2	2	5
PLY parser 8 units	10	3	1	3	2	5	14
Bounding box interface 2 units	2	2	0	0	0	0	2
Material interface (1 unit)	2	2	0	0	0	0	2
Basic shape bbox and shape int 3 unit	3	1	0	2	0	0	3
Compute bbox for scene 13 units	8	0	0	8	0	0	8
Hit func for closed cylinder 5 units	6	5	0	0	0	1	6
infinite plane 3 unit	3	0	3	0	0	0	3
Actual hours	60	43	40	11	7	0	69
Estimated Hours	60	48	36	24	12	0	

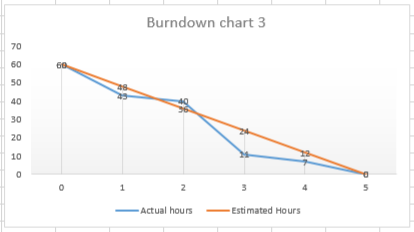
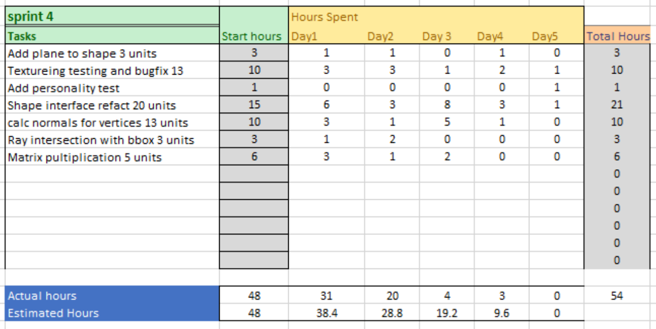


Figure 5: Burndown chart of Sprint 3

7.2.6 Sprint 4

The fourth sprint consisted of work related to a major refactoring of the program structure as well as bug fixes. The team had a conversation with one of the stakeholders who gave constructive feedback and suggestions on the overall structure of the program. Specifically, the abstraction of shapes rendered in the ray tracer program was discussed thoroughly before implementation. Thus, there were only a few items in the backlog as most of the items were dependent on the refactoring (see figure 6). As a result, many of the team members were stuck and the sprint was less productive than in the previous sprint.



sprint 4		Hours Spent					
Tasks	Start hours	Day1	Day2	Day 3	Day4	Day5	Total Hours
Add plane to shape 3 units	3	1	1	0	1	0	3
Textureing testing and bugfix 13	10	3	3	1	2	1	10
Add personality test	1	0	0	0	0	1	1
Shape interface refact 20 units	15	6	3	8	3	1	21
calc normals for vertices 13 units	10	3	1	5	1	0	10
Ray intersection with bbox 3 units	3	1	2	0	0	0	3
Matrix pultiplication 5 units	6	3	1	2	0	0	6
							0
							0
							0
							0
							0
							0
Actual hours	48	31	20	4	3	0	54
Estimated Hours	48	38.4	28.8	19.2	9.6	0	

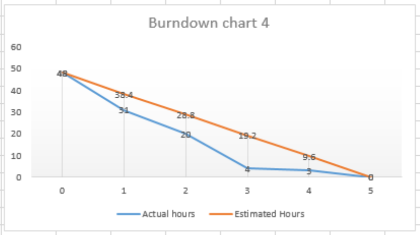
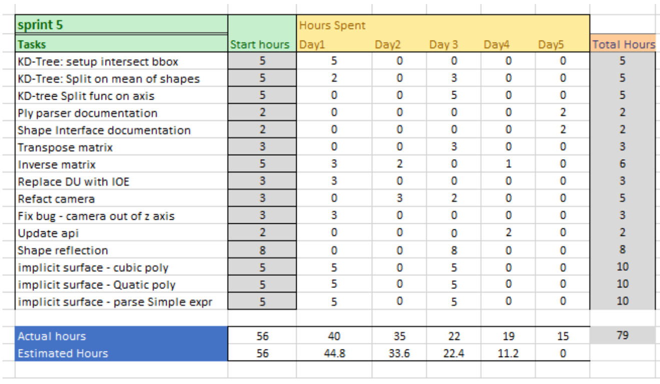


Figure 6: Burndown chart of Sprint 4

7.2.7 Sprint 5

The fifth sprint may be considered to be one of the most interesting sprints compared to the others. Since the group stumbled upon the refactoring issue of the previous sprint, many items were moved to Sprint 5 to accommodate for the lost hours that would instead be spent on implementing the missing features from the last sprint. Consequently, the group underestimated the time and size of some items which resulted in non-completed items at the end. Hence, the remaining hours left at the end of the chart (see figure 7).



sprint 5		Hours Spent					
Tasks	Start hours	Day1	Day2	Day 3	Day4	Day5	Total Hours
KD-Tree: setup intersect bbox	5	5	0	0	0	0	5
KD-Tree: Split on mean of shapes	5	2	0	3	0	0	5
KD-tree Split func on axis	5	0	0	5	0	0	5
Ply parser documentation	2	0	0	0	0	2	2
Shape Interface documentation	2	0	0	0	0	2	2
Transpose matrix	3	0	0	3	0	0	3
Inverse matrix	5	3	2	0	1	0	6
Replace DU with IOE	3	3	0	0	0	0	3
Refact camera	3	0	3	2	0	0	5
Fix bug - camera out of z axis	3	3	0	0	0	0	3
Update api	2	0	0	0	2	0	2
Shape reflection	8	0	0	8	0	0	8
implicit surface - cubic poly	5	5	0	5	0	0	10
implicit surface - Quatic poly	5	5	0	5	0	0	10
implicit surface - parse Simple expr	5	5	0	5	0	0	10
Actual hours	56	40	35	22	19	15	79
Estimated Hours	56	44.8	33.6	22.4	11.2	0	

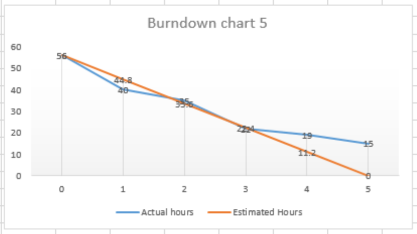
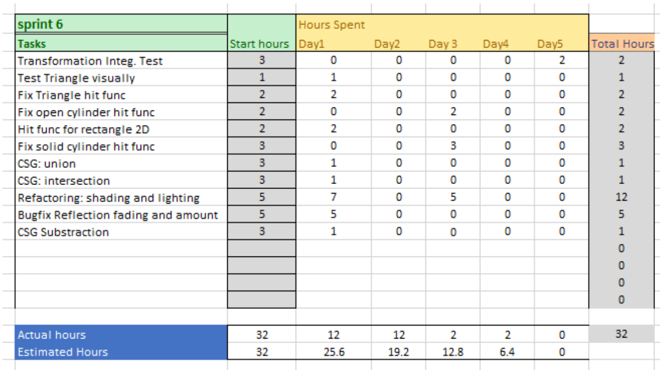


Figure 7: Burndown chart of Sprint 5

7.2.8 Sprint 6

The sixth sprint consisted of completing previous uncompleted items as well as introducing a couple of new features. (see figure 8).



sprint 6		Hours Spent					
Tasks	Start hours	Day1	Day2	Day 3	Day4	Day5	Total Hours
Transformation Integ. Test	3	0	0	0	0	2	2
Test Triangle visually	1	1	0	0	0	0	1
Fix Triangle hit func	2	2	0	0	0	0	2
Fix open cylinder hit func	2	0	0	2	0	0	2
Hit func for rectangle 2D	2	2	0	0	0	0	2
Fix solid cylinder hit func	3	0	0	3	0	0	3
CSG: union	3	1	0	0	0	0	1
CSG: intersection	3	1	0	0	0	0	1
Refactoring: shading and lighting	5	7	0	5	0	0	12
Bugfix Reflection fading and amount	5	5	0	0	0	0	5
CSG Substraction	3	1	0	0	0	0	1
							0
							0
							0
							0
Actual hours	32	12	12	2	2	0	32
Estimated Hours	32	25.6	19.2	12.8	6.4	0	

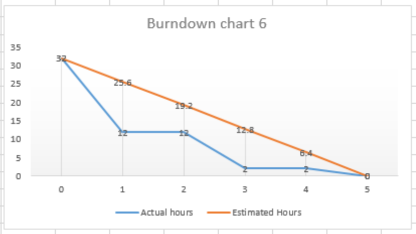
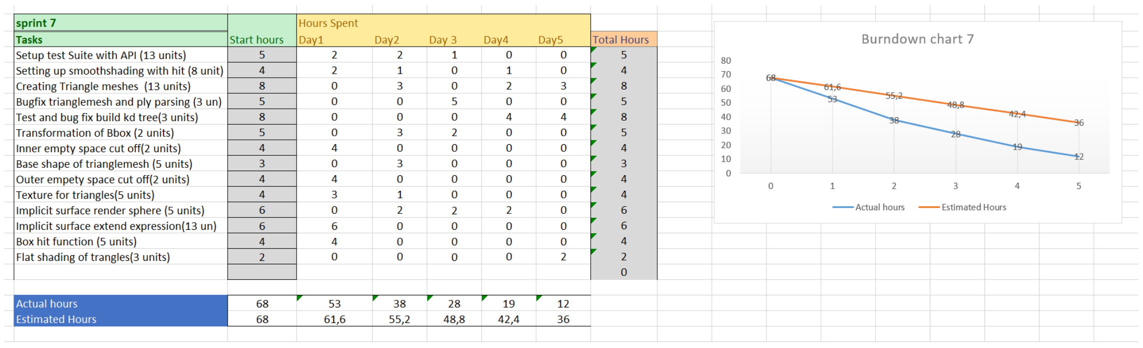


Figure 8: Burndown chart of Sprint 6

7.2.9 Sprint 7

The seventh sprint consisted mainly on bug fixing and code optimization. These tasks were manly trivial and did not introduce any spikes, hence the steady burn down.



sprint 7		Hours Spent					
Tasks	Start hours	Day1	Day2	Day 3	Day4	Day5	Total Hours
Setup test Suite with API (13 units)	5	2	2	1	0	0	5
Setting up smoothshading with hit (8 unit)	4	2	1	0	1	0	4
Creating Triangle meshes (13 units)	8	0	3	0	2	3	8
Bugfix trianglemesh and ply parsing (3 un)	5	0	0	5	0	0	5
Test and bug fix build kd tree(3 units)	8	0	0	0	4	4	8
Transformation of Bbox (2 units)	5	0	3	2	0	0	5
Inner empty space cut off(2 units)	4	4	0	0	0	0	4
Base shape of trianglemesh (5 units)	3	0	3	0	0	0	3
Outer empety space cut off(2 units)	4	4	0	0	0	0	4
Texture for triangles(5 units)	4	3	1	0	0	0	4
Implicit surface render sphere (5 units)	6	0	2	2	2	0	6
Implicit surface extend expression(13 un)	6	6	0	0	0	0	6
Box hit function (5 units)	4	4	0	0	0	0	4
Flat shading of trangles(3 units)	2	0	0	0	0	2	2
							0
Actual hours	68	53	38	28	19	12	
Estimated Hours	68	61,6	55,2	48,8	42,4	36	

Figure 9: Burndown chart of Sprint 7

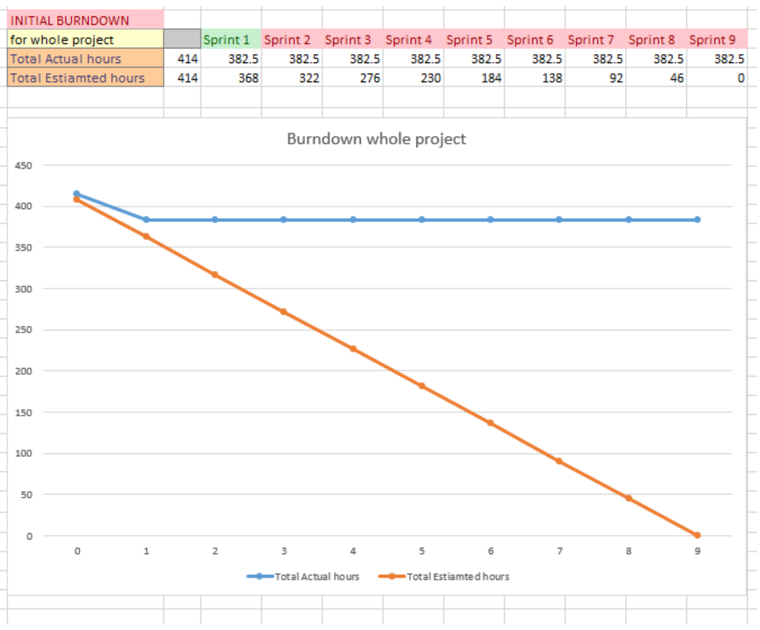
7.3 Remaining Sprints

We have not included the last two sprints as they occur after the deadline of this report. However, the last few sprints consist mainly of feature freeze, bug fixing, code optimization and report writing.

7.4 Full Estimates versus Initial Estimates

Considering the bigger picture, the project was expected to take roughly 415 hours to complete during the initial phase. This number was derived based on the expected time between 40-45 hours that a student is required to work in the course. This figure is multiplied by the number of sprints throughout the project. That is nine weeks in total. The actual estimations are relatively close compared to the estimated hours. One can see that the burn down progress in time is mostly on par or below with the estimated times even though the group had stumbled upon major difficulties midway due to the overall project structure.

Another aspect to look at is which days the team were most productive. Taken point of departure in each burndown chart for each sprint, one can clearly observe that on day one and two (Tuesday and Thursday) the chart has two major drops on the actual hours. That may fundamentally by explained be the fact that the team has to meet and discuss things before starting on the work as opposed to other days where team members were able to work individually. As a result, we have gradually understood the importance of having the team meet up for daily scrum meetings during the sprint.



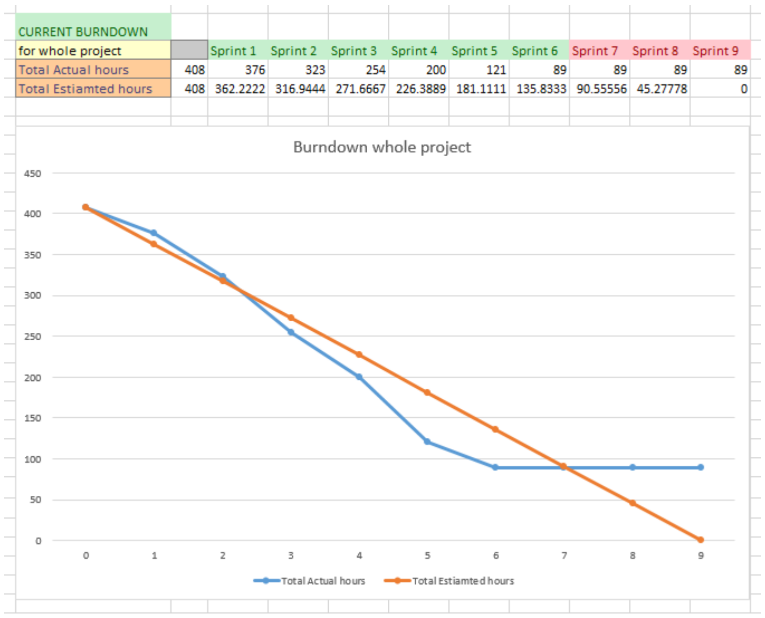
INITIAL BURNDOWN										
for whole project		Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5	Sprint 6	Sprint 7	Sprint 8	Sprint 9
Total Actual hours	414	382.5	382.5	382.5	382.5	382.5	382.5	382.5	382.5	382.5
Total Estiamted hours	414	368	322	276	230	184	138	92	46	0

Figure 10: Burndown chart of whole project - Current

7.5 Sprint Review and Retrospective

At the end of the sprint, the group at times reviewed and quality-controlled the implemented code by creating pull-requests on GitHub for each item that was to be included in the potential release. Another member would then review the work, and based on what was made, decide whether to approve the work or not.

If work was not approved, one was to fix the code based on the feedback and new requirements. If the item could not be fixed in a reasonable amount of time, the work would not be included. By way of example, the data structure used for saving objects had to be divided into additional sprints. Also, the group experienced a situation where it had to refactor the whole structure based on the decided abstraction of shapes to ensure that the program would be maintainable and easy to extend in the future. As for the retrospective, the team discussed the work process. By way of example, the team decided to increase the amount of hours spent together on Tuesdays and Thursdays. This discussion took place at the end of sprint 5 as the deadline was nearing and was based on the burndown estimations at that point of time.



CURRENT BURNDOWN										
for whole project		Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5	Sprint 6	Sprint 7	Sprint 8	Sprint 9
Total Actual hours	408	376	323	254	200	121	89	89	89	89
Total Estiamted hours	408	362.2222	316.9444	271.6667	226.3889	181.1111	135.8333	90.55556	45.27778	0

Figure 11: Burndown chart of whole project - Current

7.6 Sprint Limitations and Constraints

This section clarifies some of the limitations and constraints the group experienced and had to make decisions about, including the importance of meeting up weekly, but also defining the sprints.

Since the availability of each member and teacher assistants differ, the practice of SCRUM and its sprints had to be altered accordingly. Thus, the group only met formally on Tuesday and Thursday starting from 9 am to at most 3 pm. Sprint planning, review, and retrospective were performed on Tuesdays at the start of each sprint. The remaining days was spent on development work that was either done at home or in sub-groups.

This difference of the practice, taking from a SCRUM standpoint, is deemed to be wrong as each day in a sprint requires each member to meet up and work collectively.[24] However, this notion is perhaps more reasonable within a professional cooperation where each member can work full time and not within an academic institution in which students have to participate in other courses as well. As a consequence, daily stand-ups were only performed on these two days. The group was required to give status, but also elaborate on what was going to be worked on the next remaining days.

One can thus discuss whether the team did practice SCRUM or not. If one is to strictly follow the rules and guidelines according to SCRUM, then this is not SCRUM, but rather a methodology consisting of elements from SCRUM. If the team had to follow SCRUM, exactly every team member had to meet up every day within the five days of sprint, and the reviews should have been done more consequently.

Nevertheless, the group performed well according to the burn down charts. All together, this approach to SCRUM and sprints worked in this project, but one might consider following the SCRUM guidelines differently in other projects based on their scope, size and requirements.

[24][Sutherland, 2013, p. 10]

8 Reflection

Many choices were taken in regards to the process we chose to follow, not all of them being true to the SCRUM way of doing things. In this section, we reflect upon decisions we took, both in regards to SCRUM, but also in other aspects of the process.

8.1 Modifications in SCRUM Practices

We strove to follow the SCRUM practices as best we could while still considering where these practices did not make sense to spend time on. This is the case with the review, which we skipped numerous times. The reason behind this was based on the fact that we all checked each others' code occasionally. Also, we often helped each other out with problems related to the implementation. Therefore, we already had many looking through the code, which ensures high quality - since this is on of the main reasons behind the review, we decided to spend more time on development.

8.2 Velocity Estimation and Initial Planning

One of the biggest challenges we had was the velocity estimation. This was extremely hard for us especially in the beginning, since we did not know much about the application domain and were actually asked to start the development even before we had covered all the theory behind the project. Add to this our lack of experience with the F# language and with programming in general, and this seemed an almost impossible task. Therefore we decided not to do the poker planning for the first few weeks, but we started doing it once we had become used to the programming language and the application domain. We found that it actually did help in the sprint planning, even though our estimations were sometimes quite far from the effort it really took. Sometimes we needed to take tasks from the product backlog in the middle of a sprint because the estimation was too low, and at other times we ended up spending two sprints on a task that was estimated to take one.

Another approach would have been to spend more time on research before we started on the development. However, even after our first theory lecture, our professors set the requirement that we should implement something that could be shown on a screen already by the next week. While we tried to stick to the Scrum practices, we did not put as much effort into the initial planning and research as we might have done. Had we had the time, the initial planning could have helped us a lot in keeping track of our progress and knowing when it was time to put more working hours into the project. But still, our lack of experience with the programming language and the missing lack of a goal would have made this difficult - we did not receive the requirements in the form of an API until a few weeks within the projects. Of course, it was possible to start the work from what the professors had talked about during the theory lessons, but what we were concretely aiming for with our code we did not know yet.

8.3 Transparency

As mentioned in the discussion, the transparency was brought up a lot due to the SCRUM practices we applied. The daily stand-up meeting made everyone aware of what everybody was working on, how far they were, and if they needed help to overcome certain problems.

Particularly in this project, where so many things depended on each other, it was crucial to keep track of each others' work. At times where we did not coordinate and communicate properly, we ended up waiting for each other because one task depended on another task to be finished. Defining a critical path would have solved this problem, and even though it would have been tedious to draw, it would have saved many idle hours in the end and helped us schedule project activities.

8.4 Challenges in the Group Cooperation

Due to the variety of people and the fact that we were all on roughly the same level of programming skills, we cooperated well. The decision to develop using pair programming was agreed upon instantly as we all had great experience with this. But this also comes with a disadvantage: we all worked at home on our own from time to time, either due to interest or because the amount of work we needed to do exceeded the two days we put aside each week. This would mean that the other person responsible for that area would be unknowing of the new changes unless this was communicated clearly from the one changing it. Also, the quality control that pair programming provides would be lost.

8.5 SCRUM versus Previous Project Cooperation

The SCRUM approach to this project, compared to earlier projects, contains both advantages and disadvantages. It does, however, seem that the advantages far outweigh the disadvantages. We found that the things that SCRUM helped us with were;

- giving us a better overview of what we had to do and at what time this had to be done, by using product and sprint backlogs as well as velocity estimation. One thing that we could have done better was the initial planning as discussed above. Compared to the earlier projects where we have spent up to 14 hours of work per day in the last few weeks before the deadline, the structure of SCRUM have helped a lot to prevent this.

- giving clear guidelines as to how to approach a project. While the other projects have been managed without any specific methodologies, having a set of guidelines to stick to forced us to maintain an overview and keep a steady progress.

- providing strict deadlines for each sprint. Of course, there were sprints where not everything was implemented, but having this deadline make people work more focused and try to reach the goal within this time as opposed to our earlier projects, where the only deadline was the hand-in, which is the reason why all the tasks piled up in the end.

- preventing idle time for developers by finding the critical path. We learned the value of this even more so because we did not do it and ended up spending more idle hours than was necessary.

- preventing misunderstandings by using small steps and quick feedback (through e.g. the daily stand-up meeting). This was a problem on an earlier project where a lot of work was wasted on a feature that should not have been implemented due to a lack of communication within the group.

One thing that we found tedious was the many hours spent on these meetings - not so much the daily stand-up meeting, but the retrospective and the sprint plannings. It takes practice to know how much to tell at a daily stand-up meeting, as well as knowing when a discussion has gone too far and needs to be stopped. Thus, we once spent half an hour discussing whether to meet at 9 or 10 on Thursdays. To a developer who just wants to get his hands on the code, this time might seem wasted, even though most of our meetings were very beneficial to all of us.

8.6 Sprints

When it comes to the length of the sprints, we considered making two-week sprints from the beginning. However, after talking to the professors responsible for the API, we decided that sprints of one week would be better as the API was to be a subject of constant change. These short sprints have accommodated this very well and worked in almost all the sprints. During one sprint, we hardly managed to get any PBI's done as we encountered a problem with the structure of the whole program, which prevented many of us from implementing new features. For this, a two-week sprint would have been better, as our release did not contain any considerable changes, but this was an unforeseen problem.

The benefit of doing it this way, especially in the beginning, was that we could update the sprint backlog each week. Initially, we had a hard time estimating velocities, and the probability that we put too few tasks on the sprint backlog was big. The short sprints accommodated this problem very well, too.

8.7 SCRUM Master and the Product Owner

As opposed to a real-world project, we did not have an actual product owner. We pointed out one in our group who took care of the responsibilities of a product owner to keep the backlog groomed, but in the end we all added to the backlog as we deemed necessary. This was partly a good thing as it is the developers who know the best what needs to be done, but also partly a bad thing as it caused some confusion. Sometimes, tasks were duplicated, creating confusion as to whether this was actually done, and at other times tasks were moved because we had realized that it was far too early to start on the implementation of these, which is not in the nature of SCRUM. However, as long as the group was notified of this, we saw no problem in this as long as nobody had assigned himself to the task.

When it comes to the SCRUM master, we had an individual in our group with previous experience with SCRUM from his workplace, and it was, therefore, evident that he be the one to make sure we followed the appropriate practices. He did a good job at keeping us on track at the meetings and made sure that everything necessary was said, even though the discussions could still get out of hand occasionally.

9 Conclusion and Final Thoughts

In general, we have found that the Scrum methodology has been an excellent way to develop this particular project. Further, it provides the team with many benefits despite the problems connected to it. The team has learned a lot of valuable lessons from the practices that were left out, like the initial planning and finding the critical path. The initial planning and tools such as the Gantt chart and critical path will be considered in our next projects in order to schedule activities, get an overall overview of project deadlines and assure that we spend our time as productively as possible. However, we experienced that these planning steps were difficult to integrate into this particular project due to the unknown domain, new programming language and lack of experience with SCRUM and project planning in general.

In regards to our cooperation, we implemented various practices ourselves, which worked very well for us in this particular project. The incremental releases in the form of subversions helped us continuously maintain a stable version of the product. Specifically, practices like pair programming made the process of problem solving faster and made the Sprint Review activity less relevant.

SCRUM was deemed useful for this project due to the extensive occurrence of changes and unknown properties of the domain, leaving it hard for us to make any initial planning and estimates throughout the lifetime of the project. However, according to the burn down charts our sprints were successful even though our velocity estimation was not always quite on spot.

The biggest take away from using SCRUM in this particular project was arguably the communication and project structure improvements. Even though we did not use all aspects of SCRUM stringently, it helped us communicate within the team.

Arguably, SCRUM could be utilized for any project where communication is important, but in our case, it did not help focusing on the overall project objectives and the bigger picture in which other process models like the Waterfall model might have been considered. We think that the traditional Waterfall approach might still be used to understand long-term objectives while the agile scrum approach is suitable to facilitate communication and identify short-term wins.

We strove to use SCRUM as described in the SCRUM manifesto, but have chosen to modify it to our needs instead of following it as a recipe step by step that gave rise to a trade-off between productivity and quality assurance of which productivity was deemed most important.

Altogether, SCRUM has been a useful framework for this project, and we will most likely benefit from it when gaining even more experience in future projects.

Bibliography

atlassian. Git Workflow. https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow, 2016. [Online; accessed 16-May-2016].

Mike Cotterell Bob Hughes and Rajib Mall. *Software Project Management*. McGraw-Hill Higher Education, 2011. ISBN 0071072748.

George Dinwiddie. Feel the burn - getting the most out of burn charts. http://idiacomputing.com/pub/BetterSoftware-BurnCharts.pdf, 2009. [Online; accessed 16-May-2016].

Jr. University of North Carolina at Chapel Hill Frederick P. Brooks. Essence and accidents of software engineering.

Bob Hughes and Mike Cotterell. *Software Project Management*. McGraw-Hill Higher Education, 2009. ISBN 0077122798.

Sarah Beecham John Noll and Ita Richardson. Global software development and collaboration: barriers and solutions. *Vol. 1, No. 3*, 2010.

leanmanufacturingtools. What is Lean? http://leanmanufacturingtools.org/34/lean-manufacturing-definition-2/, 2016. [Online; accessed 16-May-2016].

Tutorials Point. Scrum Backlog. http://www.tutorialspoint.com/scrum/scrum_artifacts.htm, 2016a. [Online; accessed 16-May-2016].

Tutorials Point. Scrum Roles. http://www.tutorialspoint.com/scrum/scrum_roles.htm, 2016b. [Online; accessed 16-May-2016].

Tutorials Point. Scrum Roles. http://www.tutorialspoint.com/scrum/scrum_user_stories.htm, 2016c. [Online; accessed 16-May-2016].

Tutorials Point. Scrum Events. http://www.tutorialspoint.com/scrum/scrum_events.htm, 2016d. [Online; accessed 16-May-2016].

Mountain Goat Software. Velocity Range. https://www.mountaingoatsoftware.com/tools/velocity-range-calculator, 2016. [Online; accessed 16-May-2016].

Ken Schwaber & Jeff Sutherland. The scrum guide. 2013.

Paolo Tell. Agile estimating and advanced topics in agile planning, 2016a. System Development and Project Organization (BSUP), IT University of Copenhagen.

Paolo Tell. Scrum - a detailed overview, 2016b. System Development and Project Organization (BSUP), IT University of Copenhagen.

Mind Tools. Belbin's Team Roles. https://www.mindtools.com/pages/article/newLDR_83.htm, 2016. [Online; accessed 16-May-2016].

Wikipedia. Waterfall Model. https://en.wikipedia.org/wiki/Waterfall_model, 2016. [Online; accessed 16-May-2016].

10 Appendix

Daily Scrum Meetings - Logbook

Meeting 29 marts 2016

Topics:

- SCRUM (design design of first sprint)
    - Roles
    - PBI
- Meeting hours
    - Tuesday and thursday
    - Weekends? (oh no you DIDN'T)
- Version Control using Git
- Development
    - Individual?
    - Pair programming
- Wishes for development tasks in project
- Implementation
    - TDD (hand out tests)
    - Camera

SCRUM

We will use trello to keep track of work and backlog in SCRUM
We have decided for sprint cycles in between 1-2 weeks
Initially, we will start off with sprint cycles of 1 week
We will have a daily SCRUM meeting for 15 minutes each tuesday and thursday at 9 AM

1 week sprint: getting the basics up and running is quite simple and the initial tasks fit a 1 week sprint very well

Scrum Master: Thor
Product Owner: Jesper (professor) and internal substitute in group is Maria
Team: Thor, Daniel, Emil, Dennis, Nicoline, Maria

Review, retrospective and planning will be held on fridays 14:00PM-15:00PM

Meeting hours
We will meet each tuesday and thursday 8:30AM-14:00PM (+2 hours) PM
We will have daily SCRUM meetings 8:45-9:00
The first two initial daily SCRUM meetings will be held after the lectures from 8:30-11
Homework: setup initial project, report and logbook in Github

Github conventions

Create

Master branch: latest working release (stable)
Developer: latest new releases (unstable)
Other branches: new features and tasks implemented by individual team members

Dennis will show how to use branching in Git on Thursday 31 March
Nicoline will show how to use Overleaf for Latex on Thursday 31 March

Sprint Backlog

1) Setup initial project in GitHub
2) Setup report and logbook in Overleaf
3) Tutorial on Github and Overleaf
4) Make initial Product Backlog Items

Development Style

Individual and pair programming

Work Topics

Implicit surfaces (all shapes described through an equation, ordinary): Thor, Nicoline, Dennis

Triangle meshes (parsing, Ply, HIIT triangle, bounding box): Emil, Nicoline, Daniel, Dennis
    Parsing: (you shall not parse!)

Optimisation (parallel, kd-tree, bounding box, quad tree): Thor, Emil

Object composition (HIIT, combining objects through union and intersection): Emil, Daniel

Textures and Reflection (shadows, light, color): Dennis, Nicoline, Maria, Daniel

Object modification (scaling, rotation and skewing): Thor, Maria
Lir: Daniel

Wrap up

Thursday: initial project is setup, tutorial on GitHub and OverLeaf, small coding tasks (optional)
Feedback from teachers

1st week goal:
SCRUM goal: what are we to do according to Backlog and who wants to do what (idea)
Development goal: get a sphere up and running in a scene
Newton Raphson (do a derivative) and Ray tracer implementation is omitted from assignments

Talk to Professor and TA soon about theory

F# testing: write things into interactive, does not have to be regression tests, show examples computing correct results

Inner derivative: x^2 derivative is 2x but you might have expression (x+3)^2 = (x^2 + 9 + 6x)’ = 2x + 6

We replace and abstract away pixels with units in our calculations (relative size based on screen size)

Value of 1 pixel has nothing to do with value of pixel to the left -> no common computation

Parallel.ForLoop is used when you fire a million rays (distribute to different kernels, pixels have nothing to do with each other, works as long as screen fits in memory) -> syntactic change -> use instead of ordinary For loop

Tip: Setup a backlog, figure out what to do in sprint (setup skeleton, delegate work, do research), challenge: not all material given (triangle mesh etc, hard to setup complete backlog)

Put something into the backlog and refine it iteratively (have a basic setup for triangle meshes, e.g. 1 working on data structures and other working on rendering, after examples do optimisation scheme and compare if it looks the same)

We have overview of what the RayTracer should do allowing you to **derive PBI**

We do not make initial estimations this week

Expect a full picture by the end of next week

Meeting 31 marts 2016

Scrum meeting

- Dennis and Daniel have created a git repo, added API and Daniel’s solution
- Nikoline has created an overleaf

Git presentation

- Basic explanation of git and how to use
    - We use 2 main branches
        - Developer branch (Contains latest feature that has not been release)
        - Master branch (Contains latest release)
            - Merge developer branch to master after every sprint
        - Merge to main branches - only working code with no compile error
    - One is to branch from developer branch if one is to work one some given part of system
- Pull requests are to be made if one is to merge into master or developer branch
- Naming conventions
    - One is to declare branches with the following notation
        - category/name
            - Eg: feature/hitFunction

Overleaf presentation

- Overleaf is a web app to write latex documents
- We use two main documents for 2nd. Year project and BSUP
- We also include logbook
    - Main documents will only include sub-documents
- One is to add a new section in the folder section in overleaf and then include it in the main document
    - Use include when adding section to main document

Meeting 1st of April 2016

DENNIS' BIRTHDAY!! <(^^<)(>^^)>

Scrum meeting

What has been done:

- Daniel has worked on scene, bit map, saving pictures and ambient light since the last meeting.
- Maria finished working on implementing squareroot, minus and division for polynomials
- Nicoline finished working on finding corners on image plane with the camera
- Emil and Thor finished the sphere hit function and a solve function for solving second degree polynomials.
- Dennis has worked on structuring the program

Plan moving forward:

- Daniel, Nicoline and Dennis will work on getting rays up and running, so the bit map Daniel has made can be put into production.
- Maria and Emil will work on hit functions for the shapes triangle and plane.

Retrospective on first sprint

- Short sprint, given we only had two days
- Hard to plan next sprint given timeframe
- Time invested into module and structure well spent.
- Early results give a boost in spirit. Early progress is proceeding well.
- Cards we could not finish, were more complex than expected.
- Should we do code reviews in order to keep up with the other sub groups work?
- Code review every second sprint ?
- Read other sub groups code from home, ask questions in half hour time span. If we cant understand it at all, the code is not well documenteted.

Planning next sprint

- Divide sprint backlog into smaller work loads. Brush it up.
- Shape interface should be implemented and finished.
- Hitfunctions should progress for generic methods and more shapes.
- Matrice functionality

Sprint Meeting 5. April 2016

Clean up
Merge **SceneAsset** into developer
Merge UnitTest example into developer

What do we want to do?
Color Modification: Dennis, Maria, Daniel

- We can make color from 3 comma separated numbers (SceneAssets.fs)
- Need to be able to multiply colors

Lightning and
Hit function: Triangle case and infinite Plane (independent of each other) - Nicoline, Dennis, Emil
Matrix addition: needed for object modification Nicoline,
Bounding box: computing the bounding box and the intersection with rays Nicoline

SCRUM Meeting 7 april 2016

What did you do last time?
Daniel og Maria: lightning and texture (color)
Everything is based on ambient light (background) and shape light
Nicoline og Dennis: done with triangle meshes
Hit function for triangles but not made triangle meshes
Cramer's Rule and solving equation with 3 unknowns
T is distance you want to find but ray should not be defined with it due to unknown
Emil og Thor: object modification

Do you have any problems?
Dennis and Nicoline: PLY file holds triangle vertices and is required to make actual triangle meshes

What are you gonna do today?
Daniel og Maria: Lightning and shadows according to directions
Dennis and Nicoline: floating point arithmetic and smooth shading
Emil and Thor: Ply parsing(1), compact representation should have two types for triangles and vertices when parsing(2), each triangle should refer to a vertex in the list, handling format (3)

Tomorrow Plans

Demo of first release and code changes to get all members back on track and know what is actually required to render a simple sphere in a 3dimensional plane

Sprint Planning has been postponed for tuesday

Emil and Thor Ply implementation design choice

- Triangles will share references for vertices in one data structure since triangle meshes may have similar vertices
- Map (more space) vs binary tree to save vertices (fast lookup but requires implement)
- What structure to use for triangles?
- New structure on vertices and triangles
    - Vertex should have x, y, z and point (used to calculate normal in hit function)
        - **Vertex of point * vector**
    - Triangle should have 3 vertices (changed in hit function)
    - Parser should count all triangles and take average of normals

Tree: tuple af int (id) og vertex
Søg på 5: sammenlign med rod og nedre subtræer

Daily Standup 8 April

What did you do last time?

Emil and Thor:

- research on ply parsing
- Hard to understand library
- Make own parser (simple files)
- Looked into data structures for efficient lookup of vertices references by different triangles
- Found website with ply file examples
- **Question: library vs custom parser using regex ?**

Daniel and Maria:

- Introduced lightning
- Lightning with directions works

Nicoline og Dennis:

- Worked on triangle meshes
- Made a triangle and hit function
- Did not look into smooth shading
- Made bounding box for triangle
- **Shape is in separate abstraction module instead of being in Hit Function**

Did you experience any problems?
Daniel and Maria: Hard to move objects
If you hit a shape, it has to be farther away than minimum distance (quick fix)
Current solution: if you move around the sphere, it disappears
Do not use Transformations atm
Question: when we manipulate object position, should we take into hit function into consideration or place object in origo and use transform to move object
Nicoline and Dennis

- require ply parsing for smooth shading
- Should calculations of bounding box be in shape?
- Make check whether bounding box is hit

What are you looking into today?
Demo and code review
Topics:
Calculations on how to set pixels
Process of rendeirng sphere (where and how in the code?)
FireRay, mkCamera, createBitmap
Triangle Hit Function and Bounding Box
Tree and Transformation
(Testing) **Question: should we use test frameworks ?**

Code Review - Developer Branch explanation of drawing a sphere in F#

Main metode:

Kamera:
- Skal bruge kamera med position, retning, zoom (afstand)
- Width med hvor mange pixels der går per unit
- 512 er hvor mange pixels du vil have tegnet op
- 10 10 ratio

Ambient light:
- Tegner baggrund i farve

Shape:
- Laver shape i 0,0,0 som er samme punkt vi kigger på med radius

Scene:
- Giver shape, ingen light, ambient light, et kamera og max reflect (tager ikke højde for endnu)

Bruger funktion **renderToScreen** inde i **Scene:**
- Henter Bitmap fra kamera med **createBitmap**
- Skaber ny form med width og height svarende til BitMap
- Henter Graphics fra form og tegner billede op i (0,0)
- Paint function bliver tilføjet til event af paint
    - Paint maler Bitmap (billedet opbygget af kamera)
- Application.Run til at vise vindue og billede

renderToFile i scene laver bitmap og gemmer den i en fil

createBitmap in Camera

Man kan ikke parallelisere to ting i en for loop
Når du sender rays afsted vil du gerne have det sker på samme tid
Vi tager width gange height (kvadratpixel) og bruger modulo til fx. For width 200 få
Man kan ikke tilgå billede fra 2 parallele tråde på samme tid (bruger locks på pixel)

1 % 3 = 1 til at tegne x pixel
1 div 3 = 0 til at tegne y pixel
⅓ = 0 // tegner hver pixel i bitmap ved at tage div på y akse
(erstatter forløkke)

x	x	x
x		
x		

generateRays
Tasks:
- Inddeler plan så det svarer til pixels i oprindelig bitmap for hver pixel
- Finder pixel position i kameravinklen og skyder afsted med fireRay

fireRay
Tager liste a shapes, liste af lights og amb er taget væk og kamera, tager point som den skal skyde ray afsted i
Tager distance som er PositiveInfinity (distance til shape er uendelig)
Tegner farve som er sort
Vi sammenligner rekursivt i fireRay

startPoint er kameraets position
Laver retningsvektor mod andet punkt
Hvis listen af shapes er tom og længde er uendelig returneres sort
Ellers returneres rød for at tegne alle shapes røde (derfor røde shape)
- Laver ray ud af direction vector og start punkt og bruger hit function på ray og shape
- Hvis den rammer, tjekker den om distancen er lavere
- Første gang er distance altid lavere end uendelig
- Hvis du har ramt object før og rammer ny skal du vide hvilken shape er tættest på os, da dem bagved ikke skal tegnes

Hvis listen af shapes er tom, må distancen være uendelig eller også må vi have ramt noget fordi distancen er mindre
Shape har nu texture med color

Ny fireRay
Matcher på shapes
Returnerer nu hitpoint og distance til shape
Hvis jeg rammer objekt og distance er mindre parser vi **getTextureColor** istedet for at hard code alt er rødt

Ny GenerateRays
Tager højde for lightning
Hvis distance ikke er uendelig så tag højde for light og color for nyt objekt

Color addition
Per pixel: Objektets farve * lyskildens farve * intensiteten
Læg farver sammen ved flere lyskilder på samme objekct
Ambient light lægger du til før eller efter

Daily standup 12/4-16

What did you do last time?

Emil and Thor has researched the PLY parser. They are well on their way to having a good idea of how they want to parse the file and into what datastructures. It should not take too long to finish.

Daniel and Maria have finished implementing the lightning and shadowing of shapes. They fixed the problem of how/when to subtract a minimal boundary value.

Dennis and Nicoline have implemented the triangle hit function and have started on calculating the bounding box for all shapes. This is only finished for the triangle, however.

Did you experience any problems?
Emil and Thor are not experiencing many problems, other than the fact that they have used mutable values, but the values should follow the functional paradigm and therefore use mutable values.

Nicoline and Dennis ran into some problems regarding dependencies between bounding box and shape.

What will you do today?
Emil and Thor will try to bring their research together and build the PLY parser - the datastructure Emil has found and the code skeleton that Thor has found.

Dennis and Nicoline need to have refactorized so that Bounding Box lies within shape. Dennis and Nicoline need to collaborate with Thor and Emil in order to figure out when the normal for triangles will be generated from the normals of their points.
We need to create a release today.

Questions for Patrick

1. Do we need to create our own tests? If so testing scale. How much are we going to test?
2. When moving implicit shapes do we translate them with a transformation or do we change the equation?
3. Can we extend on API if we want to add additional features
4. Generic hitfunction or hardcorded
5. Is it a good idea to put bounding box functions in shape or separate them
6. Is it okay to use mutable functionalities when parsing PLY files? (Counter)
7. Exam: Theory, math or code?

Meeting with Patrick - Status

We can draw a sphere and triangle
We made a camera that can draw every shape
We have implemented light on sphere

Work split
Pair programming
Divide based on dependent work
Every week we get a new task for each pair

Exam: **What** are you responsible for and **why** was it designed this way?
Possible subjects: Kd tree, parallelization, ply parser, transformation
Pick something you feel confident with and can defend at the exam
Course is not about raytracing
It is not about domain but understanding enough so you can implement it
“We used this algorithm to find roots of polynomials” -> not about math
How do you take domain knowledge and structure it in code base and divide work among yourselves (what methods)
How we implemented and why (stick to something because it is easier due to time constraints)

Report
How you organize
What are individual parts
What are hard parts
We made decisions about structure of implementation
Write down reasoning for structure and what it is (diagrams)

Testing
Important to write tests for Vector arithmetic (is the dot product of two orthogonal vectors really 0 -> sanity check of basic operations)
Confident in program
Hard to test in bigger scale -> kd tree (visual)
Check optimizations through testing(Use timer and compare) (solve poly)

Test examples
Matrices and inverse transformations
Transformations: if you compose them is it the composition or the inverse of the composition ?
Constructive Solid Geometry: intersection of two shapes that do not hit should be empty (corner cases)
Focus on low level logic

Do not need to test if parser throws an exception on wrong input

Only test low level operations used all time (intersection on rays, matrix composition, vector calculations)

- We may use Niels existing tests from functional programming course

Transformation
Place shape at 0,0,0 and transform it afterwards

API
We can extend API and ray tracer
Easiest thing is to leave API untouched and take new changed from extended API

Web sites
Resources on F# collections?
Array and single linked-list
Array is constant access
ResizeList (resizing internal array, doubles in size when need new space) -> constant access and O (n) copy time

List of shape in scene can be done as linked list, you only want to traverse whole list and not specific shape
Even in KD tree you have to traverse all and put in array with those having bounding box

Mutable datastructures: behaves functional from the outside

Hand in is in 27th May
MAKE SURE EVERYONE HAS AN OWNERSHIP OF PART OF PROJECT!

Standup 14/4

What did you do last time?
Thor and Emil has been working and finished PLY parser and is now working on Hitfunction for basic shapes.

Daniel and maria has been working on textures. (Pt spheres only) and is now working

Dennis and Nicoline has been working on bounding box for all shapes and scene. They have also created a new shape abstraction.

Did you experience any problems?
Thor and Emil had problems matching ply files (as they are using regex). No problem on run time but still wrong logic

Daniel had problems with git (where)

What will you do today?
Emil and thor is working on hit function on base shapes and planes and hit function and may work on kd trees

Dennis and Nico will be working on triangle meshes

Daniel and Maria will be working on texture for other base shapes

Dennis will merge PLY parser and bounding box

Daily Stand Up meeting 19/4

Last time?

Emil og Thor: Hit Functions for Cylinder, Disc og Plane
Dennis og Nicoline: indlæsning af fil
Sekvens kan være alt (IEnumerable)
Erstattet sekvens med array i sidste ende af parsing
Daniel og Maria: uv til alle figurer undtagen triangle (mangler baseshape)
Ønsker normal hit function og hit function til triangle meshes (Dennis og Nicoline)

Any problems?

Emil: hard edges
Define shapes based on shapes (see slides)

Today?
Dennis og Nicoline: smooth shading og triangle meshes

Retrospective 19/4

Too long meetings
Poker planning on implemented things (to get a unit for the futurue used inburndown chart)
Burndown chart og poker planning på torsdag 9-12 ish

Designbeslutning
Tag højde for liste af transformations fra Shape i hitFunc i HitFunction
Shape skal kunne holde på liste af intersections

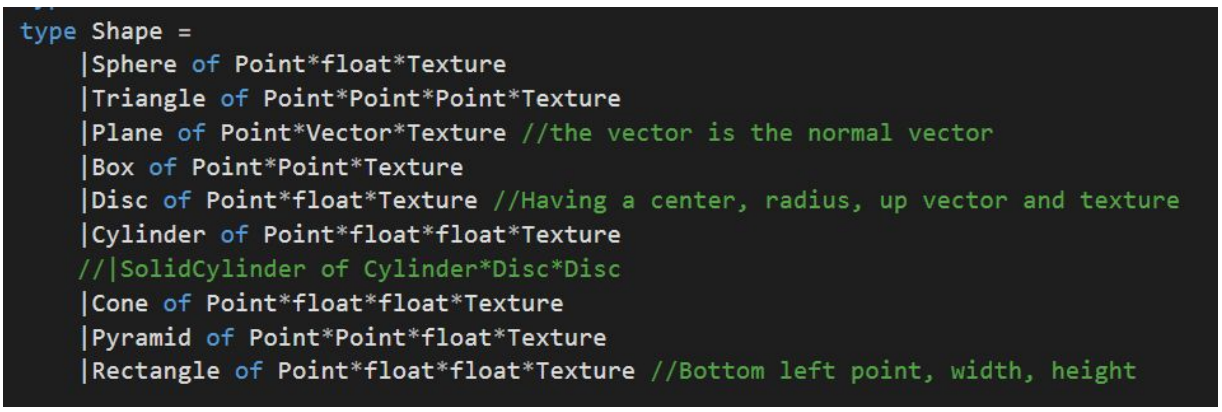
Mål
Triangle Meshes
Constructive Solid Geometry should
Reflections
Last hit functions

Sprint Planning 19/4

Poker planning

Rectangle Hit function is of unit 2
Solid Cylinder Hit function is of unit 5

19/4 april - New Design Choice: Shape Interface



```
type Shape =
    |Sphere of Point*float*Texture
    |Triangle of Point*Point*Point*Texture
    |Plane of Point*Vector*Texture //the vector is the normal vector
    |Box of Point*Point*Texture
    |Disc of Point*float*Texture //Having a center, radius, up vector and texture
    |Cylinder of Point*float*float*Texture
    //|SolidCylinder of Cylinder*Disc*Disc
    |Cone of Point*float*float*Texture
    |Pyramid of Point*Point*float*Texture
    |Rectangle of Point*float*float*Texture //Bottom left point, width, height
```

Type Shape currently requires you to change all code where shapes are used when extending. By way of example, if I add a new shape X I have to change all places where shapes are used for pattern matching in hit functions.

The solution is to implement an IShape interface that outlines the common features of a Shape and defines what Shape exists. This will allow us to follow the open close SOLID principle because the Shapes are open for extension but closed for modification in existing code.

Shapes will be curly bracket designed instead, e.g. type Triangle = { interface members ... }

Standup 21/4
What have you done?

- Open cylinder hitfunction corrected
- Refactoring the shape interface - in the future, we will use a IShape interface to split up shapes into their own files
- Tested the different hitfunctions and textures
- Calculated normals for vertices, so that we may smooth shade

Did you experience any problems?
- Merge problems has deleted some of the texture implementation for sphere

- It is hard to continue on smooth shading until the shape interface has been created and refactored.

What are you going to work on?

- Emil and Thor will continue refactoring the shape interface
- Thor will test whether the transformation implementation is working or not as there have been some confusion about what to transform.
- Maria and Daniel will try to figure out what is missing from the implementation for hit functions, textures and work to bugfix these

Standup 26/4

What have you done?

- Implicit surfaces - hvordan man løser 3. + 4.grads polynomier (research). Ikke færdigt, men godt på vej.
- Discriminate units - det eneste der bliver eksponeret i fsi er et ishape interface. Typer af shapes, men kun hvad de skal have for at kunne blive bygget.
- Refactoring - therefore, all tasks have not been done.
- Intersection with bounding box
- Calculate normal
- Smooth shading
- Units to PBI's
- Fixed bugs

Did you experience any problems?

- Instead of classes for shapes, used discriminate units
- Problems with hit function - should test to see that it renders when adding new hit function
- Multiple lights didn't work - fixed

What are you going to work on?

- BOOK THE MEETING with Patrick
- Need to finish the refactoring
- Merge

Retrospective

- We had too many PBI's
- Bad merge resulted in an ineffective sprint
- Try not to have PBI's which depend on each other
- Introduce guidelines for clean code
- Test your own code

New meeting hours

- 9-16 (stand up meeting kl 10)

Sprint planning

- Reflections
- Implement new API
- Fix transformation
- Test the test suite

Standup 3/5

What have you done?

TO:
- Transformation fixed
- use inverse and transposed matrix as described in slides
- wrote documentation for shape interface and some for ply parser (see g drive folders)
- rewrote immutable ply parser to be used later for performance comparisons

Emil: Implicit Surface Equations and expressions
- Solve cubic and quadratic expressions
- New PolynomialFormulas module

Daniel: Transformations and camera refactoring
- Refactor camera
    - Goal: better running time and more clean code
- Bugs when running program

Maria: Reflections
- Have not tested reflections
- Updated API

Dennis and nicoline: kd tree
- Can determine split points based on shapes
- Split based on dimension sizes
- Tested that scene is split correctly using drawings and interactive
- Tail recursive

Any problems?

TO: Transformation integration test, wrong hit functions

Emil: Float terminal issue

Daniel:

Maria:
- Need to test reflections
- Some API functions need to be bound to new transformation module (Thor)

Dennis and Nicoline: Heuristic, talk with Patrick, empty space split ?

What are you going to work on?

TO: transformation wrap up and start on implicit surfaces with Emil (intro), look at rendering triangle hit function

Emil: Newton method to solve n degree polynomials

- Merge polynomial into developer with solveSecondDegree

Daniel:

- Want to show refactoring changes

Maria:

- Translations need to work
- Test API

Dennis and Nicoline: finish kd tree that works for all shapes using bounding box

Sprint Meeting 3/5

Sprint Review
Product related
Maria: review kd tree
Thor: implicit surfaces and camera changes
Daniel: reflections, hit functions and shape structure

Sprint Retrospective (Process related)
Many things to integrate with each other
Only have 2 meeting days a week
Meeting hours need to be held 9-16
(Nicoline needs to meet with Amalie and Carsten thursday 2pm for 1 hour)
Too SCRUM centered -> process takes time and leaves us with less implementation

We will meet this friday 6/5 and start on the initial version of the report in BSUP
Fremover bruger vi 2-3 timer hver fredag. Hvis timen slutter 12 er det 12-15 og ellers hvis kl 14 så 14-16.

Sprint Planning

Thor and Emil: Implicit Surfaces and review of hit functions (triangle, cylinder, rectangle) and rendering triangle
Maria and Daniel: Constructive Solid Geometry and testing API
Dennis and Nicoline: KD tree

Meeting hours this week:
9-16 on Tuesdays and Thursdays
2-3 hours each week for BSUP report

Daily Standup 5/5

What did you do last time?

Emil: parser til expressions
- Parser simplifier skal udvides til square roots og division
- Udvide parser så den kan simplificere udtryk for cubic og quadratic polyomials

Dennis og Nicoline:
- Kd tree
- Mangler empty space og traversal
- Kan splitte træer ift shapes

Thor: Transformations og hit functions

Daniel og Maria: Constructive Solid Geometry works

Did you have any problems?

Emil:
- Fejl i parser til at udlede float værdi
- Jepser foreslog vi kun arbejder i expressions ved at forbedre parser (undgå arithmetic)

Nicoline og Dennis: svært at teste kd træ i interactive
Try to place kd tree in ray tracer to avoid references issues and allow testing

Daniel og Maria: shading, lightning, reflection issues, texture for triangle, rectangle, solid cylinder and disc

What are you going to do today?

Emil: kig på udvidet parser og test 3 og 4. Gradsligning

Nicoline og Dennis:
- Try to place kd tree in ray tracer to avoid references issues and allow testing
- Traversal of tree

Daniel og Maria: shading, lightnign and reflection issues with textures for new shapes

Daily Standup 10/5

What did you do yesterday?
Daniel and Maria: constructive solid geometry, corrected the light and reflection bug. Implemented the API.

Emil: Long division, almost done with a differential function. Implicit surfaces needs testing still, but it is well on the way to be finished.

Dennis and Nicoline: worked on traversal of the kd tree
Problems?

Emil: can’t test in interactive.
Dennis and Nicoline: problems with testing in interactive and circular reference problems.

Future?
Daniel and Maria: test that texture works for all shapes. Find out how to use the official test suite.

Emil: Working to finish Newton’s method, derivative and differential function.

Dennis and Nicoline: going to work on triangle mesh base shape type as well as traversal and building of kd tree.

Sprint Meeting 10/5

Sprint Review
Our code does still not show trianglemeshes and we do not know if everything will integrate well. We still need to check if each component works.

Sprint Retrospective (Process related)
The strict meeting times are not being followed. More breaks if we are to last throughout the day from 9.00 - 16.00.

Weekends are not supposed to be work days but they are also not sacred. That is work may occur if needed.

Sprint Planning

Done in Trello

Daily Standup 12/5

What did you do last time?

Emil and Thor:
- implicit surfaces
- Specifically implicit sphere
- Render a sphere today from implicit equation (goal)

Dennis and Nicoline:
- Kd tree
- Done with traversal
- Made shape abstraction for triangle mesh (holds kd tree and traverses it upon tree)
- Tested kd tree methods

Daniel and Maria:
-

Any problems?

Emil and Thor:
- Cannot debug but only print due to async
- Exchanges in expressions happen correctly but we get wrong n polynomials out upon simplifying an expression

Dennis and Nicoline:
- Kd tree code is clustered, refactor later
- MeshBuilder: do not know how to handle texture

Daniel and Maria:

- Texture: wrong things parsed
- Texture is made when creating a shape
- Need u v from ply parser

What will you do today?

Emil and Thor: goal is to render sphere from implicit equation

- Next would be to focus on roots and division in expression library

Dennis and Nicoline:

- cut off empty spaces and setup kd tree with camera
- Wrap up and integrate with raytracer

Daniel and Maria:

- Use transformations without library (double array, only required to implement transpose)
- Maria will look on textures
- Need to run test suite

Daily Standup Monday 16/5

What did you do last time?

Thor and Emil:

- Render implicit sphere using expression library and string polynomial input
- Started working on division and roots in expression tree (for n degree)
- Need to implement Newton and Sturm

Dennis and Nicoline:

- Tested kd tree and bug fixing
- Transformations with bounding box (can now transform both shapes and boxes)
- Dennis: parse triangle mesh, bug fix in hit function and ply parser
- **Demo: we can draw all triangle meshes**
- **Need to test with kd tree**

Daniel and Maria:

- Daniel implemented API
- Inserted test suite
    - **Will be easier to test product when we integrate branches**
    - **Hit function: all hit functions fail due to bugs**

What will you do today?

Thor and Emil:

- Division and roots in expression tree

Dennis and Nicoline: combine kd tree and triangle meshes

Daniel and Maria:

- **Merge branches and test suite again to determine what works and what not**

Is anything blocking you from doing your work?

Thor and Emil:

- We did not focus on wrong bugs (simplify function) and had bad knowledge of atoms and expressions
- Hit function: cannot derive negative color

Dennis and Nicoline:

- Smooth shading does maybe not work?

Daniel and Maria:

- Texture and color debugging

Sprint 8 and 9 Goals

BSUP:

1) Proofreading and grammar (“korrekturlæsning”)
2) Add sections on last 2 sprints (less PBI due to reports and other activities)
3) Write conclusion based on complete content
4) Check that content is based on common thread/theme (“den røde tråd”)
5) Add logbook as pdf reference to document SCRUM practices and give insight to daily scrum and add bibliography

Second Year Project:

1) Merge triangle meshes, kd tree, transformations, hit functions and polynomial
2) Finish KD tree
3) Make implicit surface work with Newton’s method (solve n degree) and sturm (locate distinct real roots)
4) Demo: render 4, 5 and 6th degree implicit polynomial
5) Demo Bunny within 2 minutes using ply parser and triangle meshes
6) Test:
7) Report template with brain storm on interesting design choices

Goals today

1) Merge branch for new hit functions, transformations bug fix, triangle meshes, kd trees and polynomials
    a) Merge developer into all branches
        i) Triangle mesh (Dennis and Nico)
        ii) Bbox (Dennis and Nico)
        iii) Transformation (Daniel)
        iv) Polynomial (move implicit stuff from shape into new implicit shape, Emil)
            (1) Emil and Thor will make implicit source file for implicit shapes
    b) Merge all branches into developer
    c) Build, run and push
2) BSUP report
    a) Should be proof read by Thor, Maria and Nicoline once and read by the whole group finally
    b) Proofreading today and adding references to sections
    c) Dennis makes burndown for sprint 7 and Maria and Thor explains the data
    d) Write conclusion
    e) Insert log book as reference
    f) Make bibliography and add both references in literature and foot node (reference literature with name)
3) Setup kd tree and triangle meshes and test with bunny
4) Solve 4, 5 and 6th degree polynomials
5) **Feature freeze, bug fixing and test suite the 23rd monday**

Design Choices Brainstorm

- Shape abstraction
- Hit functions and basic shapes
- Kd tree implementation and heuristics (optionally optimizations, can change heuristic in recursive call if it does not make sense to split anymore)
- Transformations and changes of abstraction
- Ply parser with mutable counter (parsing of triangles)
    - Array instead of sequence
    - Parsing of triangles
    - Mutable vs immutable
- Material lazy implementation (first calculate material when needed)
- Light: glare effect and gamma correction
- Implicit Surfaces: skipped n degree solution and developed base case rendering a sphere
- Camera: one for loop instead of two to optimize parallelization
- Test
    - Running times
    - Considered test frameworks
    - Test suite and API

Sprint Planning 17/5

Goal: demo bunny within 2 minutes using ply parser and triangle meshes