Universitat Politècnica de Catalunya

Facultat d'Informàtica de Barcelona (FIB)

Bachelor's Degree Thesis

Jordi Gil González

Analysis of the Path Tracing rendering method on CPU and GPU

Monitoring report

Computer Science Department

Bachelor's Degree Thesis Director:	Chica Calaf, Antoni
Study programme:	Computer Science
Specialization:	Computer Graphics

Academic Year 2019/2020

March 17, 2020

Contents

*

CHAPTER 1

CONTEXTUALIZATION AND SCOPE OF THE PROJECT

1.1 Introduction

Nowadays, the images generated by computers are very present in both a work and entertainment environment. The creation of realistic images through computer programs has become a necessity. Industries like the film or video games need algorithms capable of representing the real world in a virtual one and, whenever possible, in the shortest possible time.

The study about methods that allow rendering realistic images is not new. Between the early and mid 70s, the first papers about the simulation of light and colours over the surface of three-dimensional models began to be published. To understand how these methods work, we have to keep in mind the common representation of 3D models.

To represent a 3D model it uses a ”polygon mesh”, commonly known as a triangle mesh because, the triangle, is the most used polygon. This mesh consists of a set of vertexes connected through edges and creating faces. For all face, we can define an orthogonal normal vector. In Figure 1.1 we can see an example of a 3D model represented by a triangle mesh.

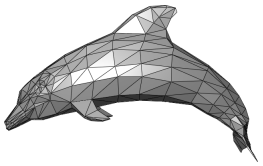


Figure 1.1: Triangle mesh example. Source: Wikipedia

Returning to lighting calculation methods, the simplest is the *Flat Shading*. This method only uses one of all the vertexes that make it up and her normal to determine the colour of each face of our mesh. In a mesh represented by triangles, it's commonly using the centroid of the triangle. The colour it's interpolated for each vertex. Every face is computed independently and this produces a visual difference result between continuous faces. In Figure 1.2 we can see an object rendered using this method. We might think that adding more vertexes improve the results, but is not the solution because when more vertexes have our meshes, more memory is required and the problem would not be solved. If we zoomed in at the 3D model, we would see the same effect (known as Mach bands) (Lotto et al., 1999, pp. 5245–5250).

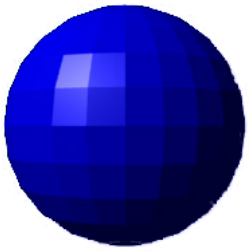


Figure 1.2: Example of *Flat Shading*. Source: Wikipedia

In the smooth shading methods, the colour change between pixels instead of faces. The result is a smooth transition of colour between adjacent faces. In 1971 Henri Gouraud presents us in his paper *Continuous Shading of Curved Surfaces* (Henri, 1971, pp. 623–629) the Gouraud Shading. With this method, we can add more continuity to the shading, unlike the Flat Shading. The great improvement concerning the method presented above is that it does not require a high-density mesh to simulate greater continuity. For each pixel it's intensity is determined by interpolation of the intensities defined at the vertex of each polygon.

- For each vertex, a normal is defined as the average of the normals of the polygons to which said vertex belongs.
- Through the using of some lighting model, e.g. the Phong reflection model, the intensity of each vertex is computed using the normal taken in the previous point.
- For each pixel, the intensity it's interpolated on every vertex to get his intensity.

As we can see in Figure 1.3, the results are notably higher, however, it does not represent the specular highlights. These would be a problem if they appear in the centre of a face.

Later, in 1975, Bui Tuong Phong in his PhD thesis (Phong, 1975, pp. 311–317) introduces to us the Phong Shading. In the method presented by Phong, instead of calculating the

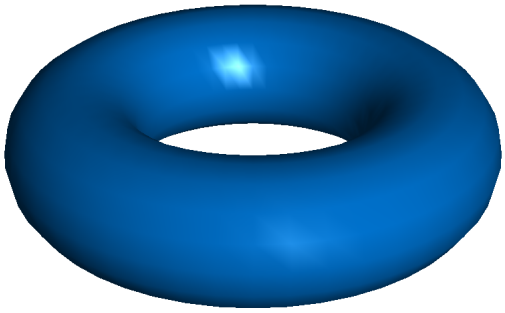


Figure 1.3: Ejemplo de *Gouraud Shading*. Source: Wikipedia

intensity at the vertex, first of all, the normal is defined, interpolated and normalized for each pixel and then using some lighting model the final intensity is determined. Computationally, this method is expensive regarding the others presented, because the calculation is made at fragment (pixel) level.

Phong mention in his paper published by the ACM (Phong, 1975, p. 311) that his goal was not to simulate the reality, but rather to add more realism:

> "*In trying to improve the quality of the synthetic images, we do not expect to be able to display the object exactly as it would appear in reality, with texture, overcast shadows, etc. We hope only to display an image that approximates the real object closely enough to provide a certain degree of realism.*"

Even though these methods represented, as far as realism is concerned, an advance, they do not pretend to simulate reality. Furthermore, these only take into account ambient, diffuse and specular light. They do not take into account the indirect lighting of the scene, an important factor in creating images that produce realistic effects such as reflections.

It was not until the 80s that the first methods capable of rendering realistic images appeared. At the sixth annual conference on *Computer graphics and interactive techniques (SIGGRAPH)*, Turner White presents the Ray Tracing method (Whitted, 1980, pp. 343–349). This method is based on the Ray Casting algorithm, presented by (Appel, 1968, pp. 37–45), consists of trace rays from the observer to all pixels, on for each one, to determine the closest object. Also, once the ray hits on a surface, based on the properties of the materials defined and the light properties, the colour is computed. In addition, using texture maps we can simulate shadows.

In 1986, David Immel et al. and James T. Kajiya, researchers from Cornell University and California Institute of Technology (Caltech) respectively, at the thirteenth annual conference on *Computer graphics and interactive techniques (SIGGRAPH)* they introduced the *Rendering Equation* (Kajiya, 1986; Immel et al., 1986, pp. 143–150, pp. 133–142). This integral

equation tries to summarize in one formula how the light interacts with a surface when a ray of light hits her using a bidirectional reflectance distribution function (BRDF). This formula takes into account the number of photons from the source, the incident angle, etcetera.

Exist other methods that can generate realistic images by approximating the RE: Bidirectional Path Tracing, presented by (Lafortune et al., 1993, pp. 145–153); Photon Mapping, formulated by (Jensen, 1996, pp. 21–30); Metropolis light Transport, introduced by (Veach et al., 1997, pp. 65–76).

1.2 Contextualization

1.2.1 Context

During the degree studies, there are several subjects dedicated to computer graphics. In these subjects, we are introduced to realistic rendering methods, but beyond the theoretical introduction to these, they are never put in practice. From here comes the idea of realizing the present project, to be able to go deeper into the subject of realistic rendering and thus create an application based on what has been learned during the studies. Other aspects of computer science seen will also put into practice, such as the creation of parallel applications on both CPUs and GPUs.

As we have pointed out in the last section, rendering true-to-life images is an area of great interests in computer graphics. One of the principal goals is to be able to render images that are indistinguishable from photographs. Following this idea, being able to replicate the behaviour of light in a virtual environment is an important task. It's essential to keep in mind the global illumination of our scene to achieve that realism. The global illumination is compounded by i) direct illumination and ii) indirect (global) illumination . In Figure 4.1, we can observe a graphic representation of both types of lighting.

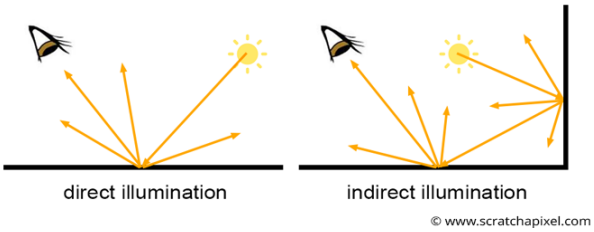


Figure 1.4: Example of *Direct illumination and Indirect illumination*. Source: ScratchPixel

i) Direct illumination is that which strikes a point from the light source.
ii) Indirect illumination is that which hits one point from the light bouncing off other points in the scene.

Retrieving what was said in the introduction, the first method capable to render realistic images was the Ray Tracing, based on Ray Casting algorithm, a technique that consists in trace rays from the eye or camera to all pixels of an image. The great novelty concerning the algorithm presented by Appel is the recursivity. The Ray Tracing algorithm emits a ray from the virtual camera across the scene to a light source. When a ray hits some surface can produce three new types of rays: i) reflection ray , ii) refraction ray and iii) shadow ray. From this point, a new ray is projected until hits one of the light sources in the scene. When tracing new rays, we can obtain effects like reflections, shadows, caustics, etcetera. If the

ray hits a transparent material like glass, the ray is projected through this to simulate the refraction rays. The principal disadvantage is the dependency on the number of polygons in the scene. The more polygons the scene will have, the more inefficient will be the algorithm.

The results obtained are not necessarily photo-realistic despite offering a high degree of realism by being able to accurately treat optical effects such as refraction or reflection. It's mandatory to do some post-processing to be able to simulate effects such as soft shadows or caustics. Rendering photo-realistic images is mandatory approximating the RE. The Path Tracing algorithm, presented by (Kajiya, 1986, pp. 143–150), is a good example of that.

The Path Tracing algorithm came up as an improvement of the Ray Tracing with the purpose to give a resolution of the rendering equation by the Monte Carlo integration. It is for this reason that the algorithm is inherently able to simulate effects such as motion blur, ambient occlusion and global illumination without any post-processing. Unlike Ray Tracing, in the Path Tracing when a ray is emitted by the virtual camera, this is traced through the scene bouncing in the objects until reaching one light source or sky or run out a limit. The significant difference between the algorithm presented by White is that the algorithm introduced by Kajiya, for each pixel, we keep in mind not only one ray but dozens, hundreds or even thousands. The random sampling is a significant aspect, this means that when a ray hits a surface, it generates a new ray in a random direction. Once a ray reaches a limit or it is absorbed by a light source, the colour is computed according to the material properties of the objects it has bounced off. This colour is added to compute the average between all rays emitted for each pixel. This random sampling produces a noisy result. The more sampling we use, the smoother it becomes.

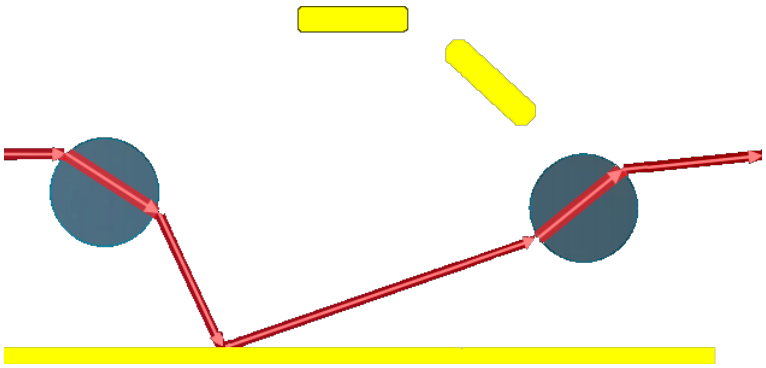


Figure 1.5: Ray Tracing

In Figure 1.5 it can be observed what is mentioned in the previous paragraph. In the Ray Tracing, the colour calculation in one pixel depends only from the primary ray and his bounces until reach a light source. However, in Figure 5 it can be observed that for each bounced, multiple rays are traced. This occurs because when a ray hits a diffuse surface, the photons are scattered in all directions.

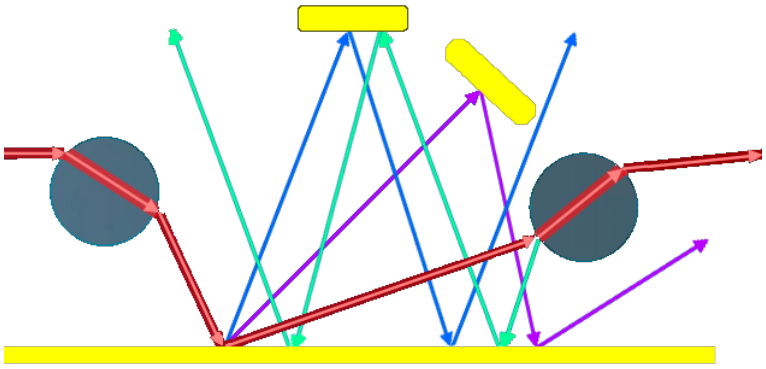
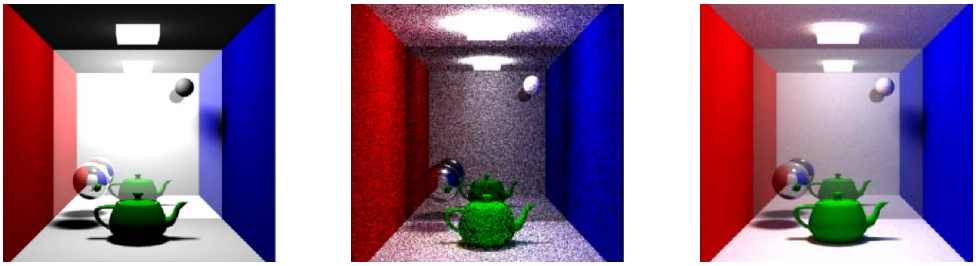


Figure 1.6: Path Tracing

The number of polygons in a scene is irrelevant for Path Tracing, the scene complexity does not affect proportionally the algorithm's performance. How we mentioned above, in the first method only one ray is traced for each polygon. However, in Kajiya's method, the ray is traced per pixel. Because each pixel is independent of each other, we can exploit the concurrency provided by CPUs and GPUs.



(a) *Ray Tracing*. (b) *Path Tracing* with noise. (c) *Path Tracing* without noise.

Figure 1.7: Comparison *Ray Tracing vs. Path Tracing*. Source: (Cassagnabère et al., 2004, pp. 23–29)

We can observe how the Path Tracing, in Figure 1.7a, produces soft shadows compared to Ray Tracing, Figure 1.7c. Also, we can see the noisy effect produced by the random sampling of Path Tracing if the number of samples is poor.

In that present project, we are going to create an application that implements the Path Tracing method trying to generate images that looks realistic. This application it's going to compute each pixel of an image given some scene. There will be three versions. The first one is the sequential version, where one pixel is computed at once and it's the base for all other versions. The other two versions are the CPU and GPU parallel versions. The reason to develop all of these versions is to be able to do a performance analysis about our algorithm both in GPU and CPU and decide which architecture is bringing more performance.

The implementation proposed by Peter Shirley in his books about Ray Tracing (Shirley, 2019b; Shirley, 2019c; Shirley, 2019a) is going to be the base to develop our Path Tracer. The main intention is to develop three applications (sequential in CPU, parallel in CPU and GPU) to analyze how is the performance in both architectures CPU and GPU.

We will not only focus on the rendering section, although this is the central thread of this project, but we will also study which acceleration techniques are commonly used to try to improve our algorithm as much as possible. That is why we will also study which is the best way to represent internally the scene we want to render following the idea presented by (Karras, 2012) in his paper.

The way we represent our scene will have a great impact on how we compute the colour of the final image. As we have mentioned above, the basis of the method is to trace rays through the scene to compute the colour of each pixel. If our representation of the scene consists of storing all the objects in a data structure such as a list or vector, ordered by order of creation, when calculating a point of the image in the worst case we will be going through the whole set of polygons of the scene to determine the final colour. Trying to render a scene that, very possibly, is composed of millions of polygons can translate into hours and hours of processing. That's why we'll make use of an accelerating data structure that allows us to represent the scene in a more clever way, so that when determining if a ray hits or not a polygon is determined in the fastest way possible.

1.2.2 Stakeholders

In this section, we are going to describe the different actors involved in a project.

Developer

This actor is in charge of planning, information search, documentation, development of the necessary software, solution of possible obstacles and problems that may arise throughout the development, and performance and analysis of the experiments. He needs to work in coordination with the director, and co-director if it exists and is the last person in charge of fulfilling the established terms.

Project director

This actor is in charge of leading the developer in case of difficulties, as well as advising on possible solutions.

Benefited users

Despite this project has not the intention to create a product on its own, it doesn't mean that there are not benefited users. Investigating the different ways of optimization using parallel architectures can be useful for many researchers or developers.

1.3 Justification

As we have already pointed in the above section, we will start from the Peter Sherly books and Tero Karras paper about Bounding Volume Hierarchies. In this project, we expect to study how we can exploit at maximum the different architectures that we have in our computers (CPU and GPU) and decide how's more efficient. We cannot compare our application with an existing one. Because the professional applications that implement the Path Tracing algorithm are made by experts with plenty of experience and knowledge that will allow increasing the performance significantly, and the author of this project does not have. Moreover, in the field of realism, it's hard to compare because a professional application includes plenty of features such as subsurface scattering or tessellation. In this project, it's impossible to reach an application with similar characteristics. First of all, the lack of knowledge and experience in the field of realistic off-line rendering. Finally, the time. Usually, a project with this scope involves plenty of specialists and need months or years to realise. A professional example of Path Tracer is RenderMan, developed by Pixar Animation Studios (Christensen et al., 2018).

For this reason, in this project, we will develop three versions of the same algorithm: i) sequential version, ii) parallel version in CPU and iii) parallel version in GPU. Therefore, we will analyze the versions with each other and decide the best architecture and version for our implementation.

Currently, there are plenty of libraries oriented to GPU programming. Libraries such as `OpenCL` and `OpenACC` provides us with portability between graphics cards from different manufacturers like `AMD` and `NVIDIA`. For this project, we decided to choose the CUDA environment developed by `NVIDIA`. `CUDA` is a parallel computing platform and API created for its graphics card and accelerators. The reason we chose this API is that being proprietary software it is more optimized for `NVIDIA` graphics cards, as indicated by (Karimi et al., 2010) and (Fang et al., 2011, pp. 216–215) in their respective papers.

It is possible that citing the `NVIDIA` graphics cards and accelerators and, keeping in mind the project's focus on rendering realistic graphics, to the lector of this Final Degree Project comes to mind the new brand of RTX cards designed by `NVIDIA`. In the beginning, the idea was to orient the project to use RTX graphics cards because they have been designed

specifically for the use of Ray Tracing in real-time. But, this idea was immediately rejected because of the high price of these. The price range varies between 350€ to 2000€ or 6000€ (for HPC). Finally, even the author of this project bought an RTX 2080 Super it was decided not to focus the project only in the new RTX technology. Exploring all the new features that these include such as RT Cores, Tensor Cores and Mesh Shaders would have meant starting the job all over again and losing a huge amount of time spent. Because it would have been necessary to learn all the necessary knowledge to develop the project. That does not imply that the graphics card should not be used. As a high-end graphics card, the number of cores it has compared to the GTX1050 (mid-range) will allow us to analyse how much better it is and whether or not it is worthwhile, in terms of cost/power ratio, to use it in applications of this type.

As we mentioned at the beginning of this section, `CUDA` is an API very optimized for `NVIDIA` hardware. This gives us an advantage because all the environments that we will use in this project have NVIDIA technology

1. Laptop - Lenovo Legion Y520 with an Nvidia GTX1050 Mobile - 4GB.
2. Personal Computer - Nvidia RTX 2080 Super - 8Gb.
3. Teaching cluster BOADA - 4 GPUs Nvidia Tesla K40c.

The use of graphics cards in different environments will allow us to analyse how our application responds in each of them and thus study how performance is when we use several cards designed for a research/professional environment, as opposed to two others designed for more ordinary use. We will also be able to see how the performance is in a mid-range graphics card (Nvidia GTX1050 Mobile) and a high range one (Nvidia RTX 2080 Super) and make a comparison between them.

1.4 Project Scope

To solve the problem presented in our project, we need an application that can render realistic-look images. As well we notice a few sections back, (Shirley, 2019b; Shirley, 2019c; Shirley, 2019a) presents the principles to create a Ray Tracing. We are going to use this to develop our main parallel application on CPU and GPU.

Furthermore, as we said in the context section, the use of acceleration structures is a significant factor in this type of applications, and they have been studying for many years by the science community, e. g.(Rubin et al., 1980). In this project, we will use the *Bounding Volume Hierarchy* (BVH), a tree structure (binary or n-ary) where each leaf represents the bounding box of each primitive; and each internal node the bounding box of his children.

There are many ways to build a BVH, but we decided to develop the version proposed by (Karras, 2012; Karras et al., 2013). We can build each internal node independently using the concurrence provided by the GPU and CPU.

1.4.1 Objectives

Main objectives

There are several main objectives that we propose in the project. The first one is to develop three applications (sequential, CPU parallel and GPU parallel) in which given a scene will render it using the *Path Tracing* method, and we will analyze the performance given by all versions developed to determine who have a better performance.

A second main objective is studying and implementing what are the best practices about memory management in a parallel application. We identified it as one of the main goals because inadequate management of it can be a serious obstacle in the development of the project.

Secondary objectives

The secondary objectives, but no less important, are the following:

1. Efficiently implement the methods presented by Peter Shirley and Tero Karras.
2. Find information about optimization techniques in the calculation of ray-object intersection.
3. Efficiently implement ray-object intersection calculations.
4. Extend the application to render 3D models.
5. Develop an interactive application.

The secondary objectives 4 and 5 will be realised according to the time remaining for the completion of the project.

1.4.2 Obstacles and risk

Although the main topics dealt with in this project have been thoroughly studied, this does not mean that the development of this project is an easy task since there are many problems or obstacles that we may face.

Main program

As we said in the previous section, the memory management it's very important. An improper manage of this may cause errors that don't allow the correct functioning of the application.

Algorithm

The algorithm that we will use computes the colour from the ray-object intersection. In the process of generating an image, these intersections are made billions of time, so an improper/inefficient implementation of the calculations can affect the performance of our application negatively.

1.5 Methodology and rigour

In this section, we're going to look at the set of tools that we're going to use during the development of the project. To be able to carry out a good development of the project, we will organize ourselves in the following form: i) Meetings every 15 days with the director to discuss the state of the project, results obtained, objectives fulfilled and determine the next steps. ii) Meetings weeklies in the final phase.

1.5.1 Methodology

The work methodology that we will follow in this project is agile. This methodology breaks development in small task to minimize the amount of planning. Each iteration, or sprint, has a set of assignments. In our project, each iteration of the development cycle will correspond with a meeting with the director until the next meeting. All meeting needs specifying the objectives to achieve to the next sprint. In Figure 1.8, we can see a scheme of how agile methodology works.

1.6 Development tools

The programming language that we will use to develop our project is English using the `OpenMP` and `CUDA` libraries.

Figure 1.8: Agile methodology scheme. Source: OpenWebinars

`OpenMP` is an API design to add concurrence to programs written in `C`, `C++` and `Fortran`. The main advantage against others with similar characteristics is the portability between different Operating Systems.

`CUDA` is a parallel computing platform and `API` developed by `NVIDIA` that allows us access to a set of instructions and compute elements inside the graphics cards and accelerators to create parallel applications.

1.7 Monitoring tools

To monitor the development of this project, we will use git to track all versions and consult them if necessary.

CHAPTER 2

PROJECT MANAGEMENT

2.1 Planning

This chapter deals with project planning and describes the tasks performed, through an action plan, to meet the deadlines relating to the delivery of the project. We will also present the resources used and analyze the possible obstacles that could modify the planning.

The project began in early July 2019 with a deadline on 13 January 2020. Finally, the delivery was postponed to April of the same year.

2.2 Task description and resources used

2.2.1 Task description

Study of concepts

Before starting the project, it was necessary to become familiar with the basic concepts relative to our project. There were a couple of meetings with the project director, during the previous semester, to define the topic of the project. Also, the author of this project enrolled in the subject of Graphics Cards and Accelerators (TGA for her acronym in Catalan) to introduce himself in the parallel computing platform `CUDA`. Since this task is carried out

outside the project plus the difficulty of determining an estimate in hours, it will not be taken into account in the planning.

System configuration

Before starting the development, we need to configure the necessary tools and guarantee her correct functioning.

To be able to develop our project it is mandatory to have installed `CUDA` in our systems. The `OpenMP` libraries are installed by default on Linux systems, so no installation is required, but it is necessary to check that they work properly. To finish, we will make use of *TexMaker*, a word processor that allows to us compile LaTeXcode to create our documentation.

The laptop and personal computer need setting up. The BOADA Cluster no needs a previous configuration because it has already been configured by the DAC (Departament d'Arquitectura de Computadors).

Project Planning

This task corresponds to the content covered by the GEP course. We can split them into three sub-tasks:

1. Context and scope of the project.
2. Time planning.
3. Budget and sustainability.

Development

Development task is the most important of all project. Includes the development of all three applications, results and documentation. We will split this phase in cycles of 14 days each one. Each cycle will have four subtasks:

- **Sequential version development:** This version is the basis for the parallel applications.
- **CPU parallel version development:** Once the sequential version has been developed, we will translate them to CPU parallel version using the `OpenMP` library.
- **GPU parallel version development:** Once the sequential version has been developed, we will translate them to CPU parallel version using the `CUDA` library.

- **Results:** Using the three applications, we will render the defined scenes and will study the performance in each one of them.

The dependencies between the different tasks are easy to explain. Until we have not the sequential version, we can't start programming the parallel versions of this one. Nor can we start a new cycle without having finished the previous one.

Final stage

In this stage, we will structure all the documentation of the previous stages and prepare the defence.

2.2.2 Summary table and estimation

Task	Time spent (horas)
System configuration	10
Project planning	133
Development	546
Final Stage	65
Total	754

Table 2.1: Hours Summary

With an estimation of 35 hours per week (3 hours from Monday to Thursday and 10 hours Saturdays and Sundays), and keeping in mind a total of 30 weeks of work, we have $35h * 30 = 1050$ hours estimated.

2.2.3 Resources used

Software

- Linux, used in all task.
- LaTeX, for documentation.
- `C++`,`OpenMP` and `CUDA`, for the development of the applications.
- `git`, used for version control and code sharing between different environments.

Hardware

- Desktop PC with the following features: i7 7700 3.6 GHz, NVIDIA GeForce RTX 2080 SUPER, 24GB RAM, 250 SSD, 1TB SSD, 1TB HDD
- Laptop with the following features: i7 7700HQ 2.8GHz, NVIDIA GeForce 1050 Mobile, 8GB RAM, 500GB SSD
- Teaching cluster with the following features: Intel Xeon E5-2620 v2 2.10GHz x2, NVIDIA Tesla K40c x4, 64GB RAM, 1TB x2

2.3 Risk management: Alternative plans and obstacles

During the development may occur some obstacles or deviations that can alter the initial planning. Thanks to the agile methodology, it is easy to manage the mishaps that may arise. The main objectives were clearly and concisely defined, and this has allowed us to know in all moments which goals have been reached and have not.

The hours planned for the different task are not static. We may need more hours, or we may even have hours left over that we can spend on other tasks.

Besides, in the case of not being able to provide different shapes to represent our 3D models such as spheres or triangles, we will only use triangles because of the versatility that these give us.

2.4 Knowledge Integration

Principally, the knowledge integrated into the project corresponds to the knowledge acquired in the Computer Science speciality, as well as some obligatory subjects of the degree and optional ones. The familiarity about realistic rendering and application of geometric transforms on 3D objects, discussed in the subject of graphics, has been required. Also, the topics studied in subjects such as Parallelism and TGA, have been very important to develop our parallel applications.

Having knowledge in programming in languages like `C` or `C++` has been important for the proper development of the project.

2.5 Identification of laws and regulations

As we said in the first delivery, the main objective of the project is to allow the author to enter the field of computer rendering, in particular realistic rendering. For this reason, there are no laws or regulations that can be applied because it is not intended to make a commercial product. Furthermore, all the materials and tools used during the project are for free use or self-created.

CHAPTER 3

BUDGET AND SUSTAINABILITY

In this chapter, we will present two significant topics, budget and sustainability. We will provide a detailed description of the costs of the project, of both material and human resources, and an analysis of how the different obstacles that may occur during the development could change our budget. We will also assess the sustainability of the project.

3.1 Budget

In this section we will make an estimate of the budget to develop the project. We can divide the budget in two big sections: i) Direct costs and ii) Indirect costs.

3.1.1 Direct costs

We divide the direct cost into the three types of resources that we have in the project: i) software ii) hardware and iii) human.

For the calculation of the amortization, we will take into consideration that the project has a duration of approximately eight months.

Software resources

All the software that we will use in our project is open source. The table 3.1 shows the cost of all software used.

Product	Price	Lifetime	Amortization
Ubuntu 18.04	0,00€	-	0,00€
LaTeX	0,00€	-	0,00€
CUDA	0,00€	-	0,00€
OpenMP	0,00€	-	0,00€
C++	0,00€	-	0,00€
Total	0,00€	-	0,00€

Table 3.1: Software resources cost

Hardware resources

The table 3.2 shows the cost of all hardware used to develop the project. The amortization takes into account an approximate duration of eight months of the project as we have specified at the beginning of this section. Approximately one year have 365 days and the lifetime of the machines used is 5 years, then this gives us a total of 1825 days of life for the computers. The project lasts from September to April therefore the total duration is 243 days.

Product	Price	Lifetime (in years)	Amortization/day	Amortization
Desktop PC	2210,09€	5	1,21€/día	294,03€
Lenovo Legion Y520	900,00€	5	0,49€/día	119,07€
BOADA	3500.00€	5	1,92€/día	466,56€
Total	6610.09€	-	-	879,65€

Table 3.2: Hardware resources cost

In the cost of the teaching cluster, BOADA, the four `NVIDIA Tesla K40c` graphics cards are not taken into account because they were a donation from `NVIDIA`, so their cost is zero.

Human resources

For each task specified in the Gantt diagram, Figure 2, we define the adequate role. The different roles are:

- **Project Manager:** Project planning, final stage and meetings with director.
- **Software developer:** Applications development and results.
- **Computer technician:** System configurations.

We can see the tasks assigned to each role and the hours estimated in table 3.3.

Role	Task	Hours
Project director	Meetings/e-mails	92h
Project manager	Project planning	133h
Project manager	Final stage	65h
Project manager	Meetings	20h
Software developer	Development	426h
Software developer	Results	120h
Computer technician	System configuration	10h

Table 3.3: Role - task.

In Table 3.4, we can see the relation between the salary in €/hour and the amount of work in hours for each role.

To determine the salary for each role, we have consulted the website (Foundation, 2019) taking into account the role itself and approximating the experience of the person (e.g. the experience that could have in a project the director, played by a senior teacher, or a developer, played by a student finishing a degree) and adding the cost of Social Security (gross income multiplied by 1,35).

Category	Gross income €/h	Gross income €/h + SS	Total hours	Estimated cost
Project director	18€/h	24,30€/h	92h	2235,60€
Project manager	15€/h	20,25€/h	218h	4414,50€
Software developer	13€/h	17,55€/h	546h	9582,30€
Computer technician	11€/h	14,85€/h	10h	148,50€
Total	-	-	866h	16380,90€

Table 3.4: Human Resources cost

3.1.2 Indirect costs

It this section, we will see the indirect costs presents in our project like the power consumption or broadband internet service.

Keeping in mind the different machines that we will use during the project, the power consumption of each one is the following listed:

1. **Desktop PC:** 320W.
2. **Lenovo Legion Y520:** 160W.
3. **BOADA:** 1500W.

The following Table 3.5 shows the link between the tasks and the machine used.

Machine	**Tasks**					**Estimated Cost**
	Configuration	Planning	Development	Results	Final stage	
Desktop PC	5h	133h	426h	40h	65h	669h
Lenovo Legion Y520	5h	-	-	40h	-	45h
BOADA	-	-	-	40h	-	40h

Table 3.5: Hours per task and machine

To estimate the cost of kW/hour, we have consulted the web portal (Group, 2019) and calculated the average price between all companies.

Table 3.6 shows the price per kW, cost and power consumption for each machine in terms of the number of hours expended in them. In *general costs* cost includes office supplies like pencils, paper, etc. and transport.

Machine	**Price**	**Units**	**Estimated cost**
Desktop PC	0,13508€/kWh	214,08kWh	28,92€
Lenovo Legion Y520	0,13508€/kWh	7,2kWh	0,97€
BOADA	0,13508€/kWh	60kWh	8,10€
Internet Service	50€	8 months	400,00€
General Costs	120€	-	120,00€
Total	-	-	557,99€

Table 3.6: Consumption cost per machine

3.1.3 Unexpected costs

In case of any deviation in the planning, we will allocate a part of the budget to all setbacks that can come up.

Category	Net income €/h	Gross income €/h	Total hours	Estimated cost
Project director	18€/h	24,30€/h	20h	486,00€
Project manager	15€/h	20,25€/h	10h	202,50€
Software developer	13€/h	17,55€/h	20h	351,00€
Computer technician	11€/h	14,85€/h	2h	29,70€
Total	-	-	-	1069,20€

Table 3.7: Unexpected costs

3.1.4 Total budget

Table 3.8 shows the total budget with a contingency of 10% for unforeseen events.

Concept	Estimated cost
Direct Cost	
Software	00,00€
Hardware	879,05€
Human Resources	16380,90€
Indirect cost	557,99€
Unexpected cost	1069,20€
Subtotal	19956,34€
Contingency (10%)	1995,63€
Total	21951,97€

Table 3.8: Total budget

3.2 Management and budgetary control

In terms of human resources, this is not an initial investment, so constant monitoring is not possible. We will only be able to compare the final cost with the budgeted cost once the project has been completed.

About material and digital resources (software), we must do constant monitoring. To carry out budgetary control, at the end of each task, we will recount the hours spent. Using this information, we can make a comparison between the real and estimated hours to calculate the deviation. In the case of a significant difference, we will have to analyse the reason for the deviation and adjust the budget estimations in the following tasks.

We will use the following formulas to calculate the deviations:

- Labour-force deviation in price = (std cost - real cost) * actual consumption of hours.
- Resources deviation in price = (std cost - real cost) * real consumption.
- Labour-force deviation in consumption = (std hours consumption - actual hours consumption) * std cost.
- Resources deviation in consumption = (estimated consumption - real consumption) * real cost.
- Total deviation in labour-force = total std labour-force cost - total actual labour-force cost.
- Total deviation in resources = total std resources cost - total real cost resources.

3.3 Sustainability

3.3.1 Self-assessment

The EDINSOST project designs a survey to self-assess our knowledge of sustainability.

I have noticed my lack of analysis in sustainability when carrying out a project. However, I'm able to recognize the causes and consequences, but not the possibles solutions that exist about it. I have rarely analysed a project or problem environmental sustainability.

Even though I'm capable of analysing a problem like this, I'm unable to understand what social, economical and environmental effects there may be in a project.

Despite understanding the need to introduce social justice, equity, or transparency into a project, I'm not setting these needs into practice. So I'm unable to estimate the real impact that a project may have on society. I know how to assess the economic viability of a project, but not how to make it compatible with the environmental and social fields.

In collaborative projects, I know and use the tools that can help all the partners. Also, I appreciate the work done by others.

In conclusion, despite to know the causes and consequences that a project has on the environmental, economical and social fields, this survey has allowed me to see that there is a carelessness in me about analysing these consequences.

3.3.2 Environmental impact

The environmental impact produced by this project remains on the power consumption from all the machines used. Has not been considered the use of any type of recycled material or material from previous projects, because we are not creating a physical product indeed.

In Table 3.4, we can see that the total hours from the human resources are 866h. Whereas a person in his usual routine consumes 0,1 kWh, the total consumption is $0,1kWh \cdot 670h = 86,6kWh$.

As we said in the last section, we use three different computers to do the experiments, but for all development, we will use only one computer. Using the desktop PC we will develop the project, his power consumption is 250W, keeping in mind the hours spent by this computer, as we can see in table 3, the total consumption is $0,25kWh * 669 = 167,25kWh$. When the project starts, we have not analysed the environmental impact to minimize it. But now having a greater perspective, we could have chosen the computer with the lowest power consumption and left the others only for the experiments (which requires less working hours and therefore lower power consumption).

According to Generalitat de Catalunya (Catalunya, 2019), in 2018 the emission of CO_2 from mains is 321g CO_2/kWh. We are consuming a total of 275,1kWh, this supposes an emission of 88,31kg of CO_2. A Barcelona - New York flight emits 1.177kg of CO_2. So, our project emits almost 7,35% of the emissions of CO_2 from a commercial flight between Barcelona and New York.

Finally, It is worth mentioning that this project doesn't offer any environmental improvement. The environmental impact that may involve power consumption depends exclusively on the company. Because the same the use of renewable energy is not the same as the use of fossil fuels.

3.3.3 Economical impact

In the last section, we have detailed all costs relatives to the project, both humans and software and hardware. Despite all the roles defined, only one person will carry on all functions (except the position of "Project Director"). The human resources cost is reduced to only two persons.

The main economic cost that we have in this project is the hardware. We could reduce this cost, maybe, giving less usage to the machines, but restricting the objectives defined. That's the reason why we think that the economic expense is adequate.

3.3.4 Social impact

Once the project finish, this will have no signification impact on society. As we said chapters back, this project does not propose any alternative to existing solutions, but rather to give the author a series of knowledge to initiate in the realistic rendering field. It is hard to determine the social impact that this project could have.

CHAPTER 4

DESIGN AND DEVELOPMENT

4.1 Introduction

Ray tracing is one of the most popular techniques for rendering realistic images of given a scene. As we said in contextualization chapter, this technique derives from the ray casting algorithm. *Ray casting* tries to find the closest object along a ray. Once the ray hits an object, the algorithm estimates the incoming light at the point of intersection and combining it with the material properties of the object, compute the final colour. A new ray could be cast to a light source to determine if the object is shadowed.

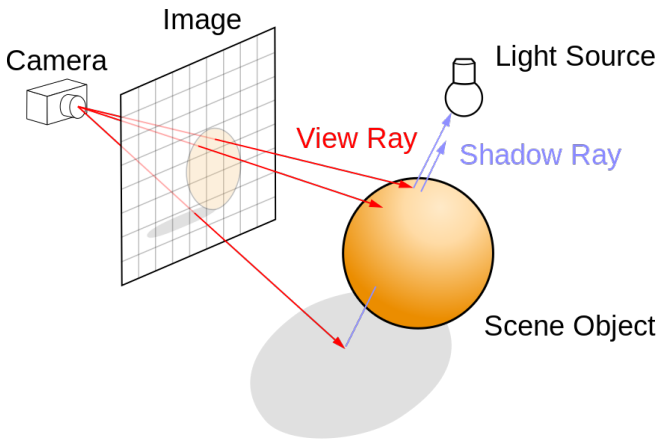
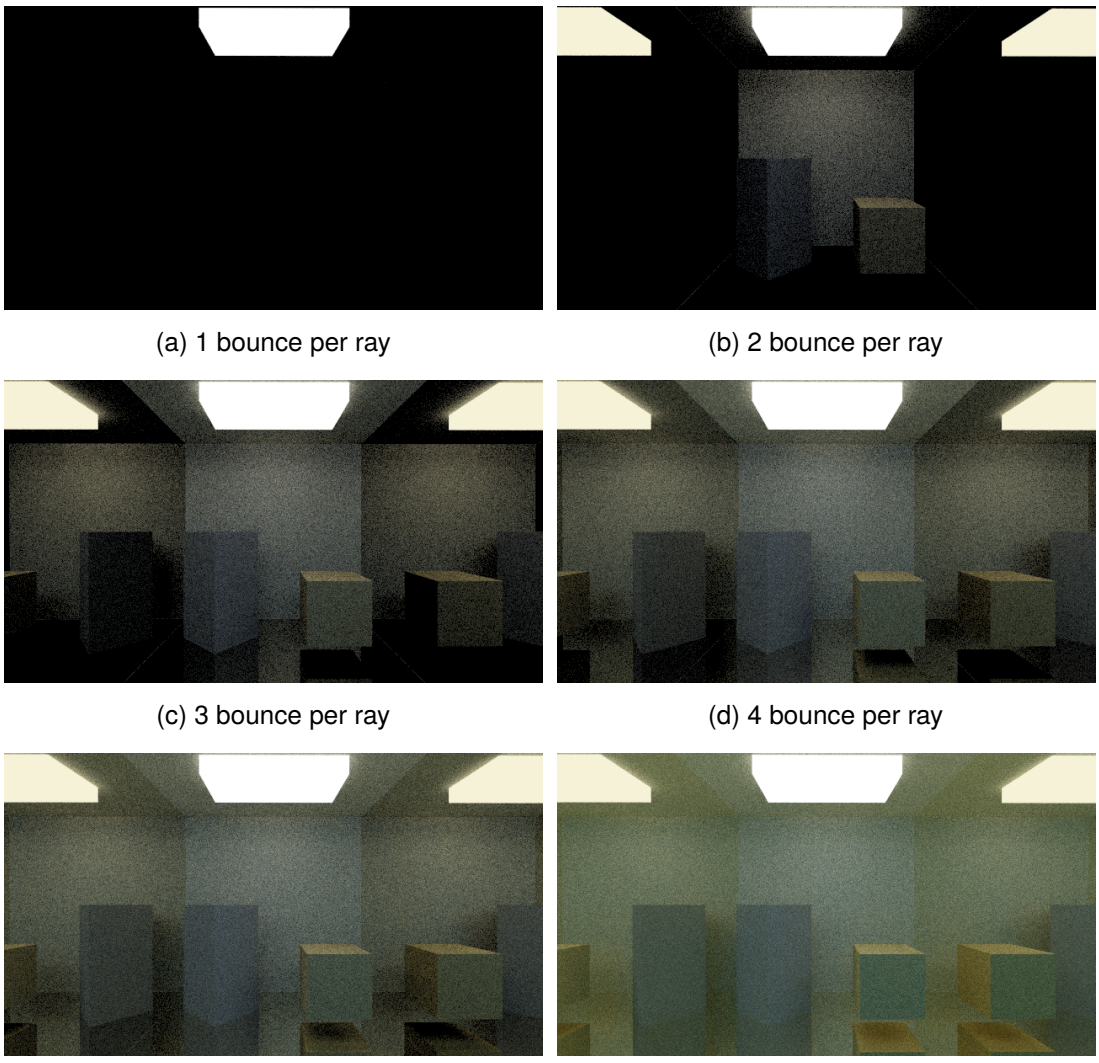


Figure 4.1: *Ray Casting* algorithm overview. Source: Ray tracing (graphics), Wikipedia

Ray tracing uses this mechanism to recursively accumulate the light contribution from

reflective and refractive objects. Materials like mirrors or polished metals produce a reflection ray, and transparent or glass objects may produce both reflection and refractive rays. This process occurs recursively, which each new ray may generate new both reflective and refractive rays. Recursion has some cutoff limit, such as a maximum number of bounces (depth). Moreover, the recursion ends when the ray hits a light source or leaves the scene. The number of bounces affects directly to the resultant image. In Figure 4.2, we can see how the more bounces, the more reflections appear in the rendered image.



(a) 1 bounce per ray

(b) 2 bounce per ray

(c) 3 bounce per ray

(d) 4 bounce per ray

(e) 5 bounce per ray

(f) 20 bounce per ray

Figure 4.2: Reflections increase with the numbers of bounces.

4.2 What is a ray?

4.3 Triangles

4.4 Materials

4.5 Textures

4.6 BVH

BIBLIOGRAPHY

[1] R. Beau Lotto, S. Mark Williams, and Dale Purves. “Mach bands as empirically derived associations”. In: *Proceedings of the National Academy of Sciences of the United States of America* (1999), pp. 5245 –5250.

[2] Gouraud Henri. “Continuous Shading of Curved Surfaces”. In: *IEEE Transactions on Computers* (1971), pp. 623 –629.

[3] Bui Tuong Phong. “Illumination for Computer Generated Pictures”. In: *Communications of the ACM* 18.6 (1975), pp. 311–317.

[4] Turner Whitted. “An Improved Illumination Model for Shaded Display”. In: *Communications of the ACM* (1980), pp. 343–349.

[5] Arthur Appel. “Some techniques for shading machine renderings of solids”. In: *AFIPS Spring Joint Computing Conference*. 1968, pp. 37–45.

[6] James T. Kajiya. “The rendering equation”. In: *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1986*. 1986, pp. 143–150.

[7] David S. Immel, Michael F. Cohen, and Donald P. Greenberg. “A radiosity method for non-diffuse environments”. In: *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1986*. 1986, pp. 133–142.

[8] Eric P. Lafortune and Yves D. Willems. “Bi-Directional Path Tracing”. In: *Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics ’93)* (1993), pp. 145–153.

[9] Henrik Wann Jensen. “Global Illumination using Photon Maps”. In: *Rendering Techniques ’96*. Springer Vienna, 1996, pp. 21–30.

[10] Eric Veach and Leonidas J. Guibas. “Metropolis light transport”. In: *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1997*. 1997, pp. 65–76.

[11] Christophe Cassagnabère, François Rousselle, and Christophe Renaud. “Path tracing using the AR350 processor”. In: *Proceedings GRAPHITE 2004 - 2nd International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia*. 2004, pp. 23–29.

[12] Peter Shirley. *Ray Tracing in One Weekend*. Peter Shirley, 2019, p. 41. URL: https://github.com/RayTracing/InOneWeekend.

[13] Peter Shirley. *Ray Tracing: The Rest of Your Life*. Peter Shirley, 2019, p. 53. URL: https://github.com/RayTracing/TheRestOfYourLife.

[14] Peter Shirley. *Ray Tracing : The Next Week*. Peter Shirley, 2019, p. 50. URL: https://github.com/petershirley/raytracingthenextweek.

[15] Tero Karras. “Maximizing parallelism in the construction of bvhs, octrees, and k-d trees”. In: *High-Performance Graphics 2012, HPG 2012 - ACM SIGGRAPH / Eurographics Symposium Proceedings*. 2012.

[16] Per Christensen et al. “RenderMan: An advanced path-tracing architecture for movie rendering”. In: *ACM Transactions on Graphics* (2018).

[17] Kamran Karimi, Neil G. Dickson, and Firas Hamze. “A Performance Comparison of CUDA and OpenCL”. In: *Computing Research Repository - CORR* (2010), p. 10.

[18] Jianbin Fang, Ana Lucia Varbanescu, and Henk Sips. “A comprehensive performance comparison of CUDA and OpenCL”. In: *Proceedings of the International Conference on Parallel Processing*. 2011, pp. 216–215.

[19] Steven M. Rubin and Turner Whitted. “A 3-dimensional representation for fast rendering of complex scenes”. In: *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1980*. 1980.

[20] Tero Karras and Timo Aila. “Fast parallel construction of high-quality bounding volume hierarchies”. In: *Proceedings - High-Performance Graphics 2013, HPG 2013*. 2013.

[21] WageIndicator Foundation. *TuSalario.es*. 2019. URL: https://tusalario.es/ (visited on 10/12/2019).

[22] Selectra Group. *¿Cuánto vale el kilovatio hora de luz en España?* 2019. URL: https://tarifasgasluz.com/faq/precio-kwh (visited on 10/12/2019).

[23] Generalitat de Catalunya. *Factor d’emissió de l’energia eĺectrica: el mix*. 2019. URL: https://canviclimatic.gencat.cat/ca/actua/factors_demissio_associats_a_lenergia/ (visited on 10/12/2019).

Appendices

APPENDIX A

GANTT DIAGRAM

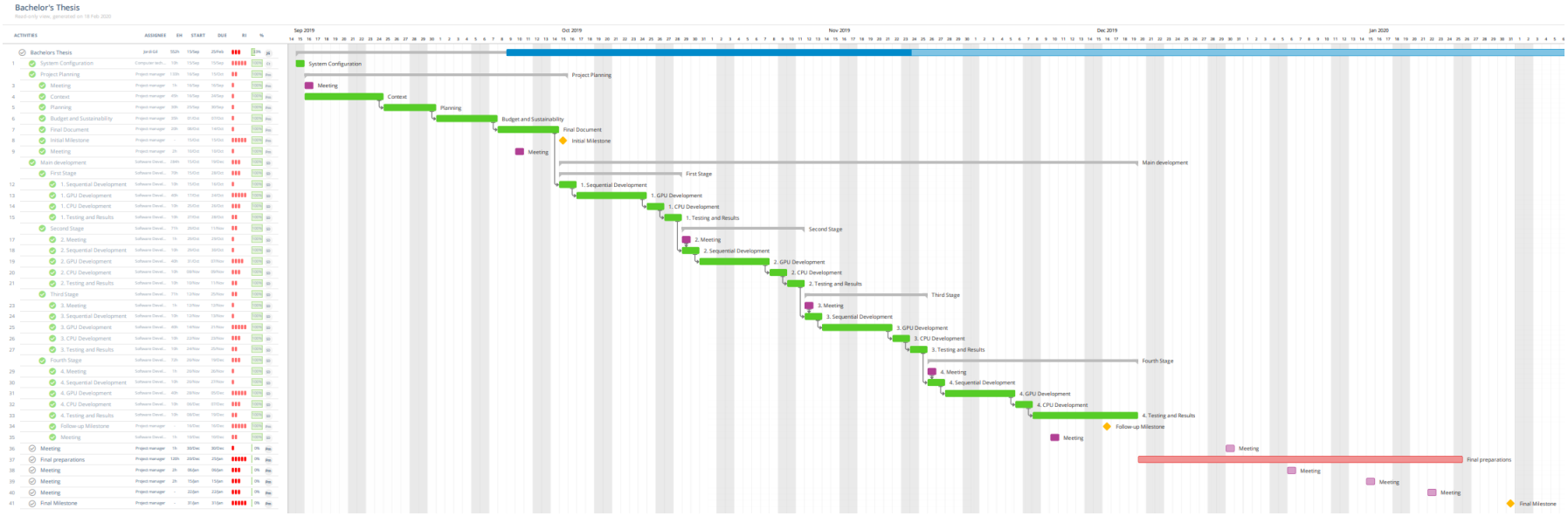


Figure A.1: Complete Gantt Chart.

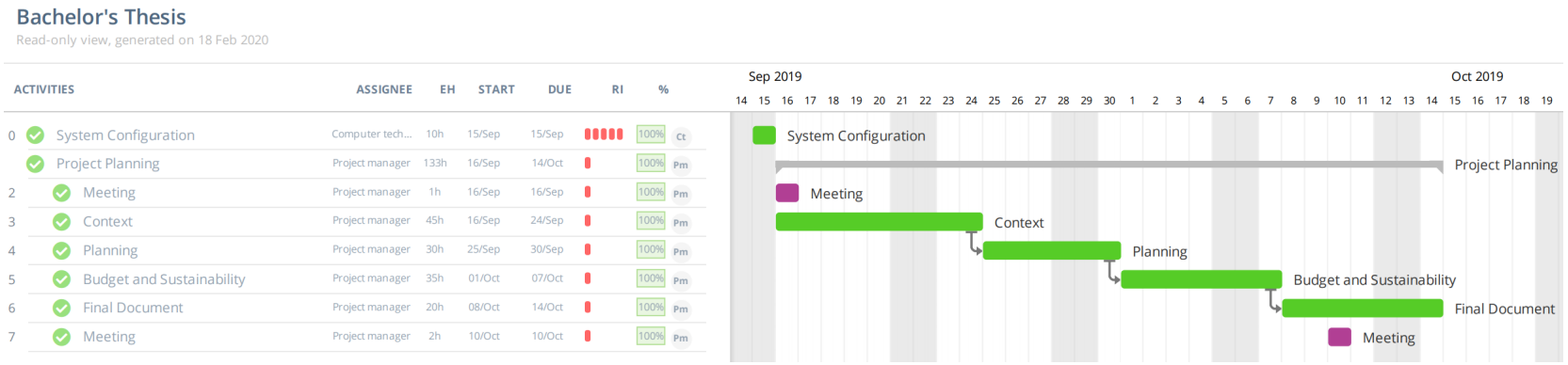


Figure A.2: GEP Stage.

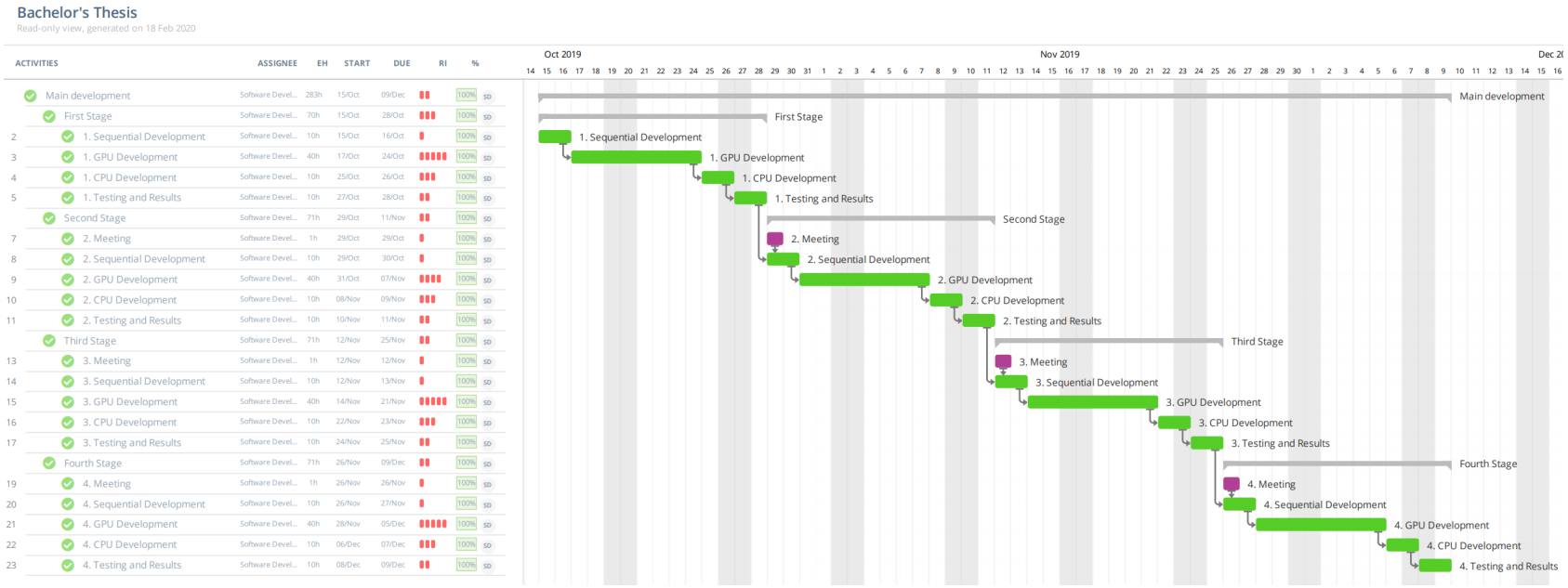


Figure A.3: Development Stage.

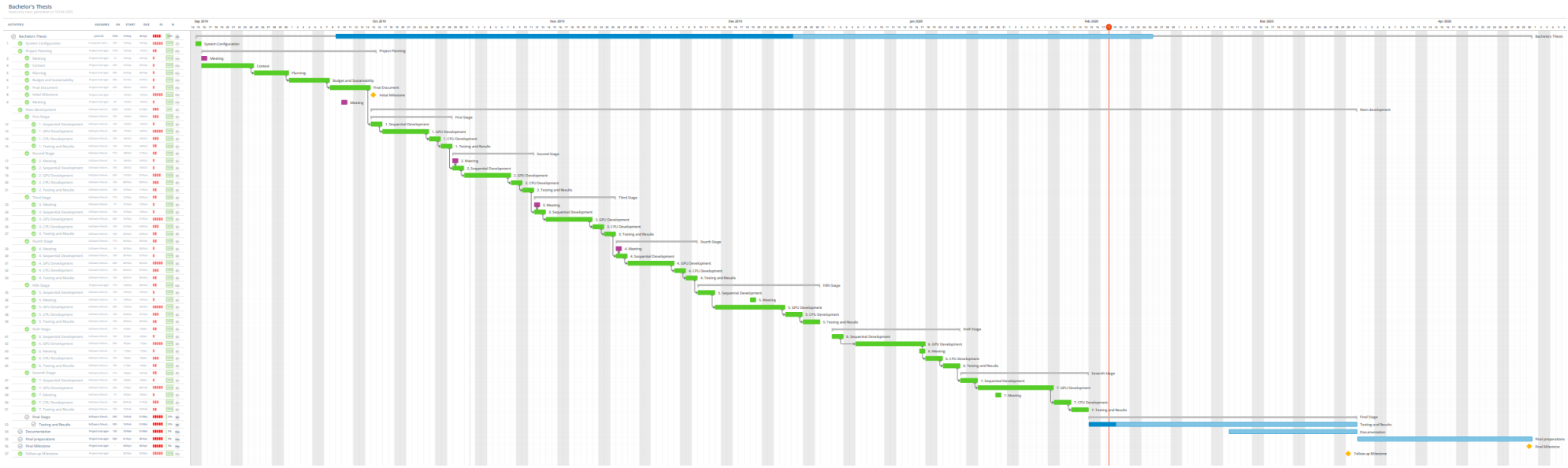


Figure A.4: Complete Gantt Chart.

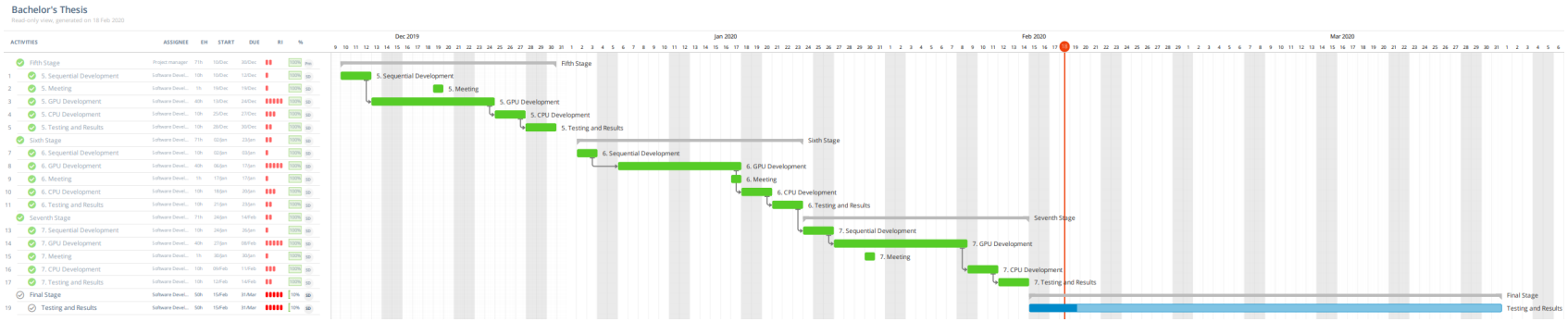


Figure A.5: Development Stage new tasks.

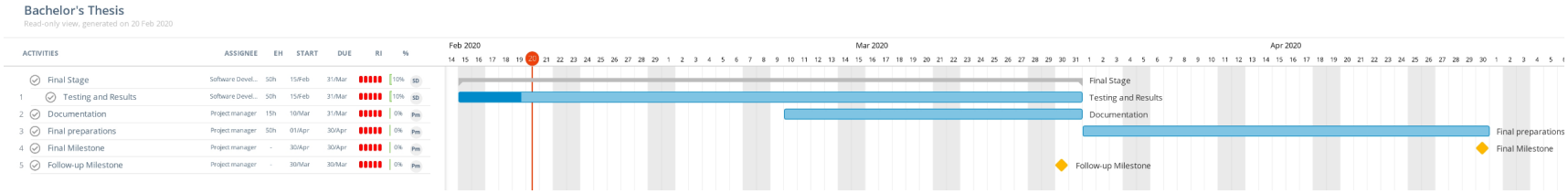


Figure A.6: Final Stage.

List of Figures

List of Tables