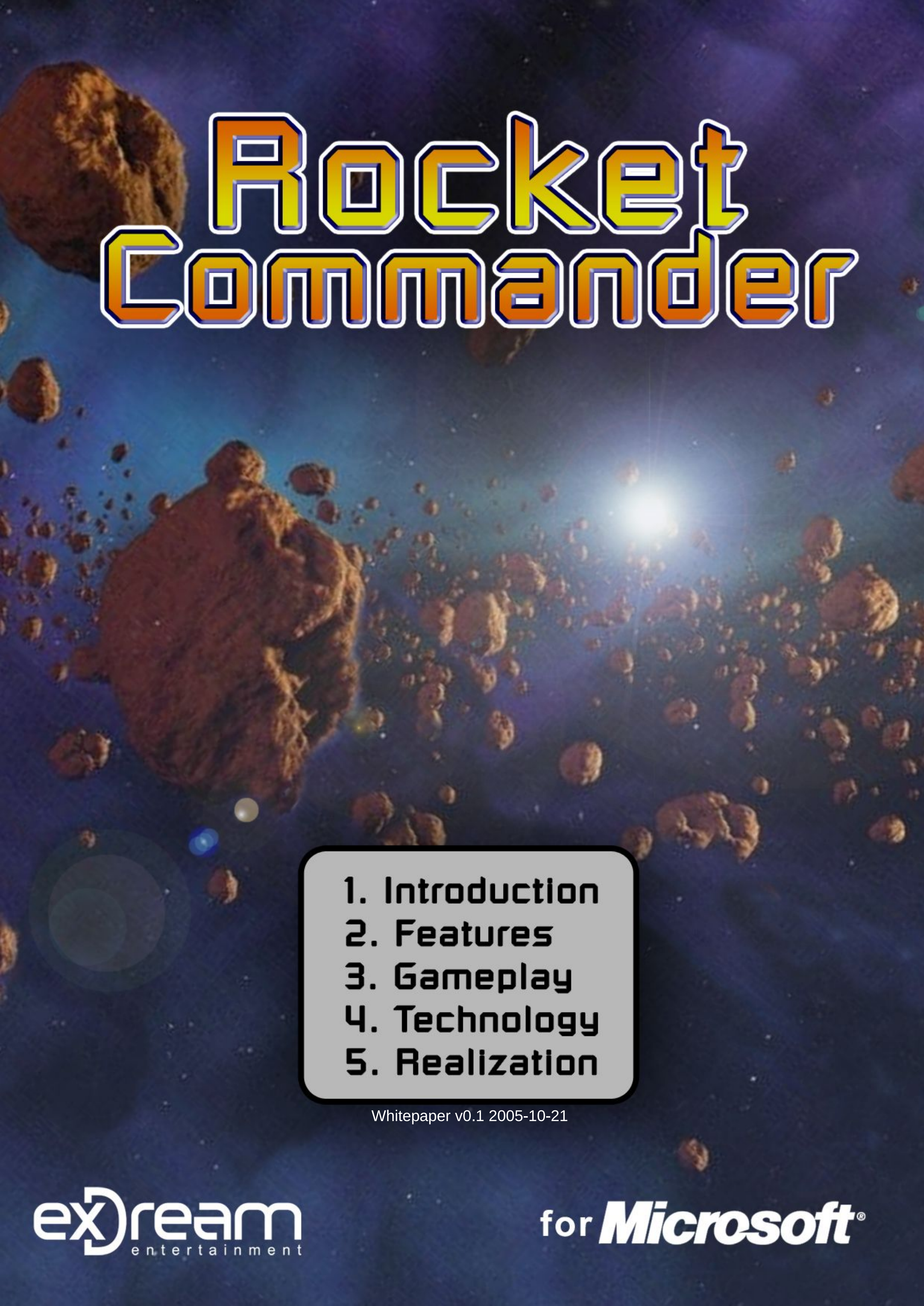
# Rocket Commander



Whitepaper v0.1 2005-10-21

exDream entertainment

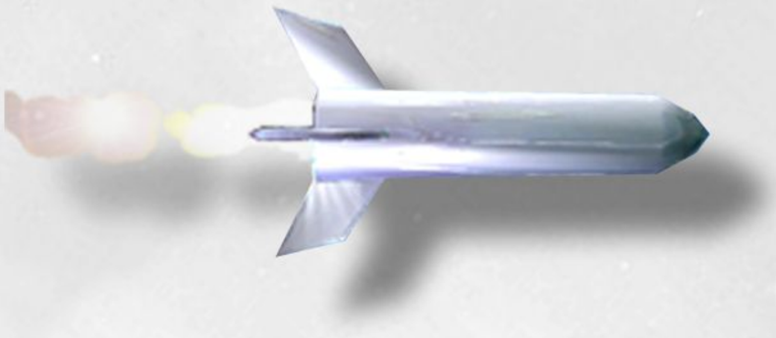
for Microsoft®

# 1. INTRODUCTION

Rocket Commander is a little beginner friendly casual 3D game. The player finds himself inside a rocket flying through a city, an asteriod field or other levels. The basic task is not to kill enemies or to solve any quests like in many other games, but to finish a level by flying through it without colliding with anything. Additionally there are special items, which can be collected to increase the health of the rocket, add additional fuel, increase the speed for a short time and give more lifes.

The enviroment is far open and the player can navigate into any 3D direction he wants to. The graphics are oriented to movies like "Star Wars" and "5th Element" (asteriod fields in the intro). To help him archive the mission objective a radar with the target location is displayed on the screen. The current health and fuel status is also displayed on the screen. Minor collisions will only damage the rocket, but bigger collisions will end fatal. The game is not only simple and beginner friendly, but because of the high speed of a rocket the game is also interessting for hardcore gamers liking a challange and training their reflections. Each player can decide for himself if he flys a safe route or flys into dangerous areas and close to big objects.

The game consists in the basic version just of a main menu, a simple singleplayer mission (in an asteroid field) and an online highscore table. More levels (city, landscape, sci-fi), more maps, multiplayer game code, more features, etc. are not planed yet, but could be implemented at a later point, see Extensions (4).



# 2. FEATURES

* Optimized for .NET 2.0 and Managed DirectX (Dezember 2005, MDX 2.0) using c#.

* Game includes Visual Studio 2005 project files (source code, help files, etc.). The sourcecode is highly commented and explains the in a compact and easy way what the game does.

* Stunning 3D effects: Advanced Shader Technologies utilizing the latest DirectX version. Includes NormalMapping (e.g. Asteroids will look really detailed), HDR Lighting (Sun and Reflections), Pre-and Post-Screen effects for colorful Sky-Cube-Mapping and post screen Glow effects and realtime Motion Blur.

* Easy gameplay, beginner friendly, but due the high speed of Rockets very exciting, thrilling and still challenging for more skilled players.

* Intuitive User Interface and Controls (very similar to shooters like Quake3 or simulators like Flight Simulator)

* Extensible interface for adding more features at a later time like network multiplayer support, more graphical effects and levels, shadow mapping, etc.

* Interchangeable item graphics and elements for the mission. Players can not only change the source code, but also put in other graphics or 3D objects using the Microsoft .X file format.

* Text events in the game are allowing a small story and showing progress, e.g. when using the game to show several milestones of a project, which have to be reached.

Image taken by the Hubble Telescope, it shows that the galaxies are not just a bunch of boring white dots. The game screen can both be very colorful and still be realistic.

# 3. GAMEPLAY

Due the short development time of only 1 month everything is cut down to the basic idea of flying a rocket through an asteroid field. All additional ideas (see Extensions) are left out of the basic game. This simplifies not only the gameplay, but also the understanding of the techniques and the code.

Games with similar elements (at least the flying part) are: Descent, Terminal Velocity, Fury, Rebell Assault, etc. The main difference in Rocket Commander is the complete freedom in the 3D space, which is also known from Space Simulators like X-Wing, Wing Commander, Homeworld, etc. However, the game is more likely to be played like the first mentioned games because you don't have to fly through empty space over very long distances. Instead the 3D world is filled with objects, all more or less located in a plane area. Since the player can freely navigate into any direction, he can also move out of the asteroid plane area. This has 2 disadvantages: First of all we will not longer move directly to the destination, thus making the overall way much longer. And most importantly we will not be able to collect any items or get additional health and fuel our rockets, which is required to accomplish the mission.

**These are the basic game elements:**

**Rocket:** This is the rocket we are flying with. In the game we will not see it except from the optional outer view (see Extensions).

**Asteroid 1**: One type of a bigger asteroid we have to avoid. Crashing into it will either reduce our health or might even kill us if we don't have much health left.

**Smaller asteroids**, the sky or effects on the screen don't interfear with the player. They are only used to help orientating and giving a better feeling about the current speed.

**Asteroid 2**: Very big asteroid, can be dangerous because it rotates around. A direct collision could instantly kill us, we might survife only collinding with a border, but this will decrease our health too.

**Collectable Items:**

**Fuel**: Gives us additional fuel we desperatly need to finish any mission. This will be most likely the most used item.

**ExtraRocket**: Gives us an additional life.

**Bomb**: Gives nuke abilities to this rocket, which causes asteroids to explode when we collide with them. Only used one time, but we can stock up multiple nukes.

**Health**: Gives us extra health to survife longer.

**Warp**: Increases the speed to 150% for a short time.

Items are exchangable with other logos like:

Microsoft® Microsoft .net

# 4. TECHNOLOGY

Rocket Commander will be programmed in c# using .NET 2.0 and DirectX 9.0c (December 2005 Edition with MDX support for .NET 2.0, which isn't avialable for VS2005 in any other DirectX version yet). The game will not only just use the latest technologies, but also show nice tricks and tips how to solve problems using .NET 2.0. One example would be the big asteroid field, which requires a lot of objects and big lists. To keep the game running at over 100 Frames Per Second the base engine must be very flexible and fast, this is where generics, anoymous methods and other new .NET 2.0 collections and features come into the game design and help keeping things simple, fast and efficient. Another big plus is using the Managed DirectX API, which helps to write very easy to read code, which executes powerful functionality like loading textures, using shaders or displaying models with just 1 line of code.

The graphics are all .dds files and .png files (for external transparent images like logos). All models (Asteroids, Rocket, Effects, etc.) are modeled with 3D Studio Max 8 and exported into Microsoft DirectX .X files. Models will not use animations for easier understanding of the underlying code.

The game will not use any engine or complicated framework. Non-game programmers would not be familar with it and this would only confuse anyone wanting to take a look at the source code. The game consists basically of this parts:

* Helper classes for Texture loading, Model loading, Form handling, DirectX handling, Controls, etc.
* Shader class to support rendering models with shader effects (only a NormalMap shader is used for this game)
* Pre and Post screen shader class to help us rendering the sky and adding cool effects like glow and motion blur.
* Main program class, which manages the game loop and handles the form.
* Main menu class for starting a game, checking the highscores or quiting the game.
* Highscore helper class to transmit highscores to the online server.
* Highscores are also saved locally, which can be viewed from the main menu
* Mission class to play the game utilizing all the other helper classes.

The game will be very small (around 5-10 MB) and can be downloaded from the web or put on CDs without any traffic or space problems.

## Possible Extensions

This section describes features, that would be cool to have, but they will most likely not make it into the game because of the short timeframe we have to develop it. Another reason for some of the features is that it would complicate such a simple game to much, both on the gameplay side and on the readability side for the code.

Since the game consists only of a couple of simple modules (helpers, main menu, highscore, mission), it can be extended quite easily. An upgrade supporting online multiplayer modes, more missions or additional game modes could be implemented quite easily. Another idea would be to help developers submitting their ideas and sourcecode by putting the project on GotDotNet or SourceForge at a later time.

Icons can be exchanged and text messages and be stored for the default mission, but to implement multiple missions and maybe a complete storyline or more game modes, the game would require more sofisticated data models and classes supporting this (multiple levels, configuations, etc.).

One of the most obvious extensions would be to implement the originally planed city missions and extending graphical capabilities with more shaders, shadow mapping, effects and so on.