Game Programming Lab 2012

Dawn of Time

Mattis Paulin / Romain Prevost / Marco Zimmermann

GAME PROPOSAL

GAME DESCRIPTION

STORYLINE

In the very dawn of time took place an explosion whose magnitude would never be encountered again. It happened 13.7 billion years ago and it shaped our universe the way we know it, the way physicists have spent centuries trying to figure out. Yet, the beginning of ages still remains a mystery, and the truth is harder to handle than most would think.

These were indeed troubled times. Using the powers of darkness, a pitiless entity going by the name of the Dark Architect, spawned antimatter at a sickening pace, in an attempt to annihilate the newly born matter. If the universe was able to survive in this boiling chaos and havoc, it was not without the help of a few brave entities, whose dedication and leading skills ultimately gave them the upper hand.

The Architects, as they were called, each had their designs and views about the way matter should form, and how it should be shaped to build the most robust structures. In the end, the strongest and slyest was able to survive, and using the strength of his dead former companions, pushed the Dark Architect back to the edges of outer space and laid the bricks of the universe we now live in.

GAME CONCEPT

In this Tower-Defense-like game, each player is one of the Architects. Two types of resources (nucleons and electrons) can be collected, the first from a shared source, the second from an individual one. Both sources should be protected at all cost from the antimatter waves thrown by the Dark Architect. To do so, players can craft atoms out of these resources. With these atoms they can build basic structures (atom by atom or by small predefined structures) and special constructions like turrets and reflectors.

Waves strike randomly and aim indifferently at all sources. Sources, as well as turrets and other special buildings, have health that they lose when hit by anti-matter. Between-player interaction is also strongly suggested, as we allow stealing unguarded piles of atoms and specific particle-induced actions.

The last man standing wins. Yet, if the common source falls, the game is a draw between the remaining players.

STRATEGY

Our game framework is designed to render an optimal strategy very hard to find. Players should find a fine balance between cooperation to protect the main source, and defending himself.

To encourage cooperation, players will be awarded more resources as they defend the main source. Stealing from other players should be rewarding but also risky.

Our main goal will be to tune the gameplay to challenge the players in their choices of strategies.

'BIG IDEA' BULLSEYE

RELATION TO THE THEME 'ATTRACTION'

Nucleons and electrons are combined together due to attraction (electromagnetism, and strong interaction). These are the building blocks for all the structures in our game. Our technical achievement will be to determine if the inter-atoms attractions are powerful enough to withstand the gravitational attraction. Finally the antimatter particles launched by the Dark Architect are attracted by the other particles, destroying the work of the Architects.

GAME CONCEPT ARTWORKS

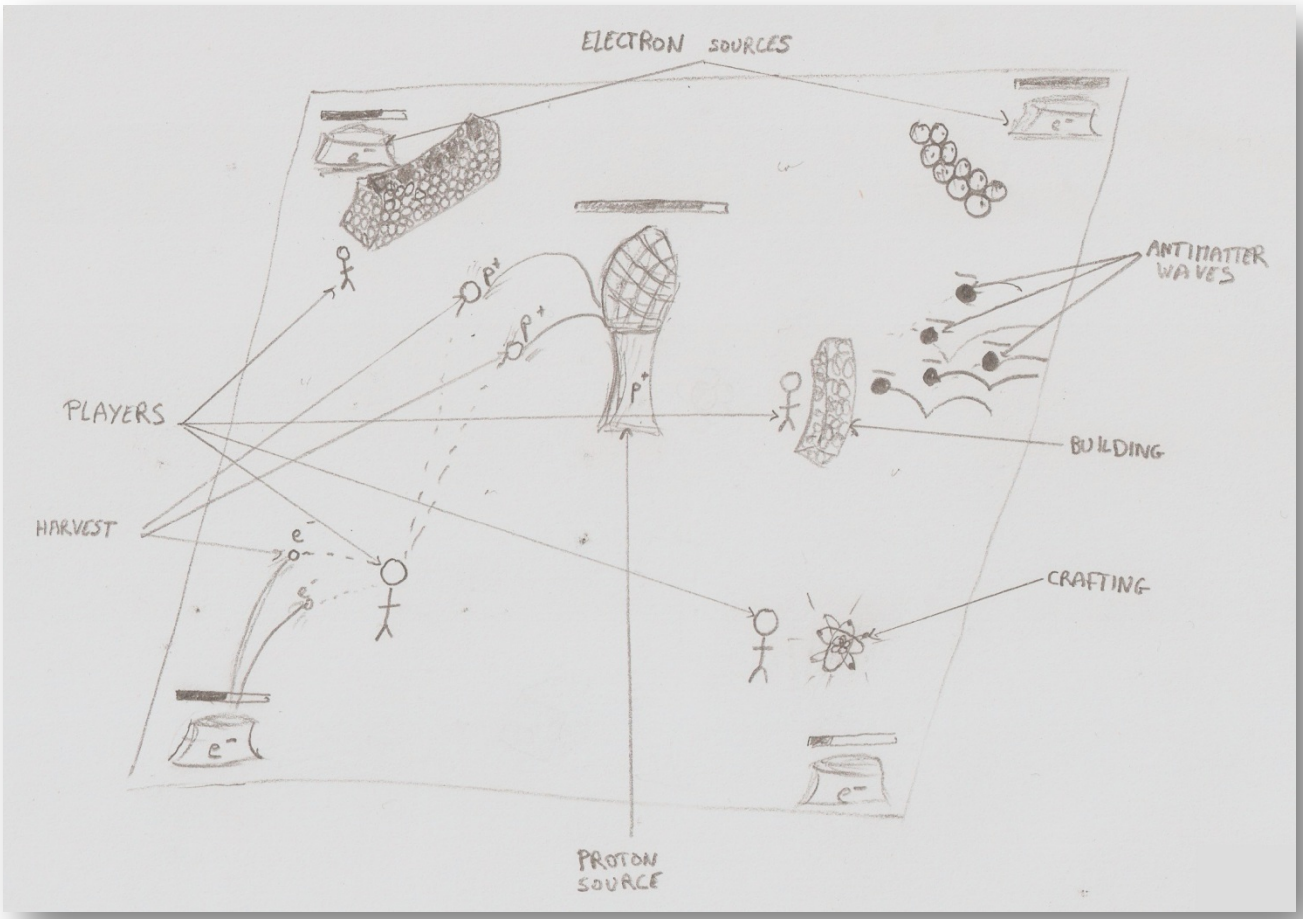


Fig 1: Map design with particles sources and other gameplay elements

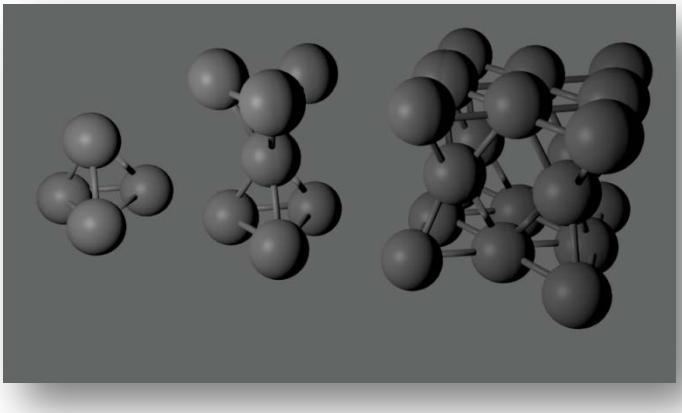


Fig 2: Concept of tetrahedral structures with atoms

FORMAL ELEMENTS

- **Players:** the game is designed for 2 to 4 players.
- **Interaction pattern:** one of the strengths of this game is the complex interaction pattern, interleaving cooperation and multilateral competition.
- **Objectives:** the main goal of the player is to protect the shared nucleon source and his personal electron source, which provide him with particles.
- **Procedures:** in order to protect himself, the player needs to create atoms with the elementary particles he collected and combine them to build more robust defenses.
- **Rules:** each atom has a particular purpose which is described later in this document. Special buildings can also be built.
- **Resources:** the primary asset is the particles and the atoms that you possess. You can use them to protect yourself and also to attack other players.
- **Boundaries:** the spatial constraints correspond to the map design.
- **Outcome:** Antimatter waves target both the shared source and the individual ones. If the shared source goes down, the game is a draw. The winner is the player whose source is the last one standing.

ATOMS DESCRIPTION

There are 10 different atoms that you can create by putting nucleons and electrons together. You can do it in the player interface really quickly. There are two categories of atoms: the ones you can use for your buildings (in green) and the ones with special effects (in red).

- **Hydrogen** (1 nucleon + 1 electron): this is like a snowball that you can throw to other players to prevent them to steal your atoms, or just to annoy them. This will also be the munitions for the turrets.
- **Carbon** (12 nucleons + 6 electrons): this is the basic atom for your constructions.
- **Oxygen** (16 nucleons + 8 electrons): it makes you move faster.
- **Aluminum** (27 nucleons + 13 electrons): this is an intermediate atom for your constructions.
- **Iron** (56 nucleons + 26 electrons): this is a strong atom for your constructions.
- **Xenon** (132 nucleons + 54 electrons): this can be used to poison another player for example if he tries to steal from you.
- **Platinum** (195 nucleons + 78 electrons): this acts as a catalyst and can be used to accelerate the process of building structures.
- **Gold** (197 nucleons + 79 electrons): this is the strongest atom for your constructions.
- **Lead** (208 nucleons + 82 electrons): this atom is less strong than gold, but it cannot be stolen by other players.
- **Uranium** (235 nucleons + 92 electrons): this is an explosive atom and will make huge damage if you throw it on constructions.

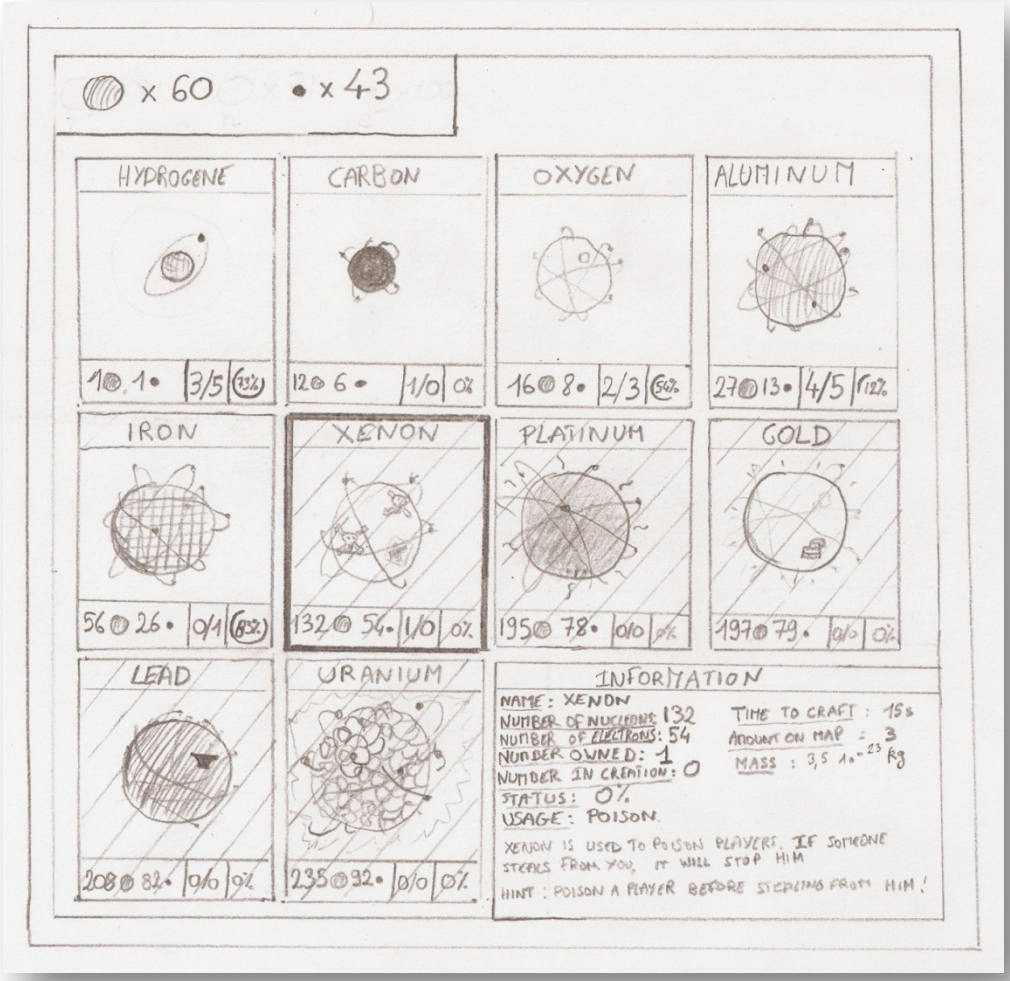


Fig 3: Help screen for atom information

GAMEPLAY

The main view for the player will be 3D. Using the gamepad the player will be able to move around the scene, to craft new atoms, to throw atoms on his buildings to fix holes, and to use special atoms (for example to oxygen run faster as described before).

The second view for the player is the building framework, in which he can create his defenses. It will basically be a top view around the player position, in which he can use his atoms to build several structures (walls, turrets…). (cf. Fig.4)

BUILDINGS FRAMEWORK

- ✓ Wall: A simple template for fast structure creation. Can be destroyed and stolen particle by particle.
- ✓ Matter turret: Defensive structure that shoots at anti-matter. Cannot be stolen and has health like sources.
- ✓ Anti-player turret: Same as the latter but aims only at sneaky players trying to steal your valuables.

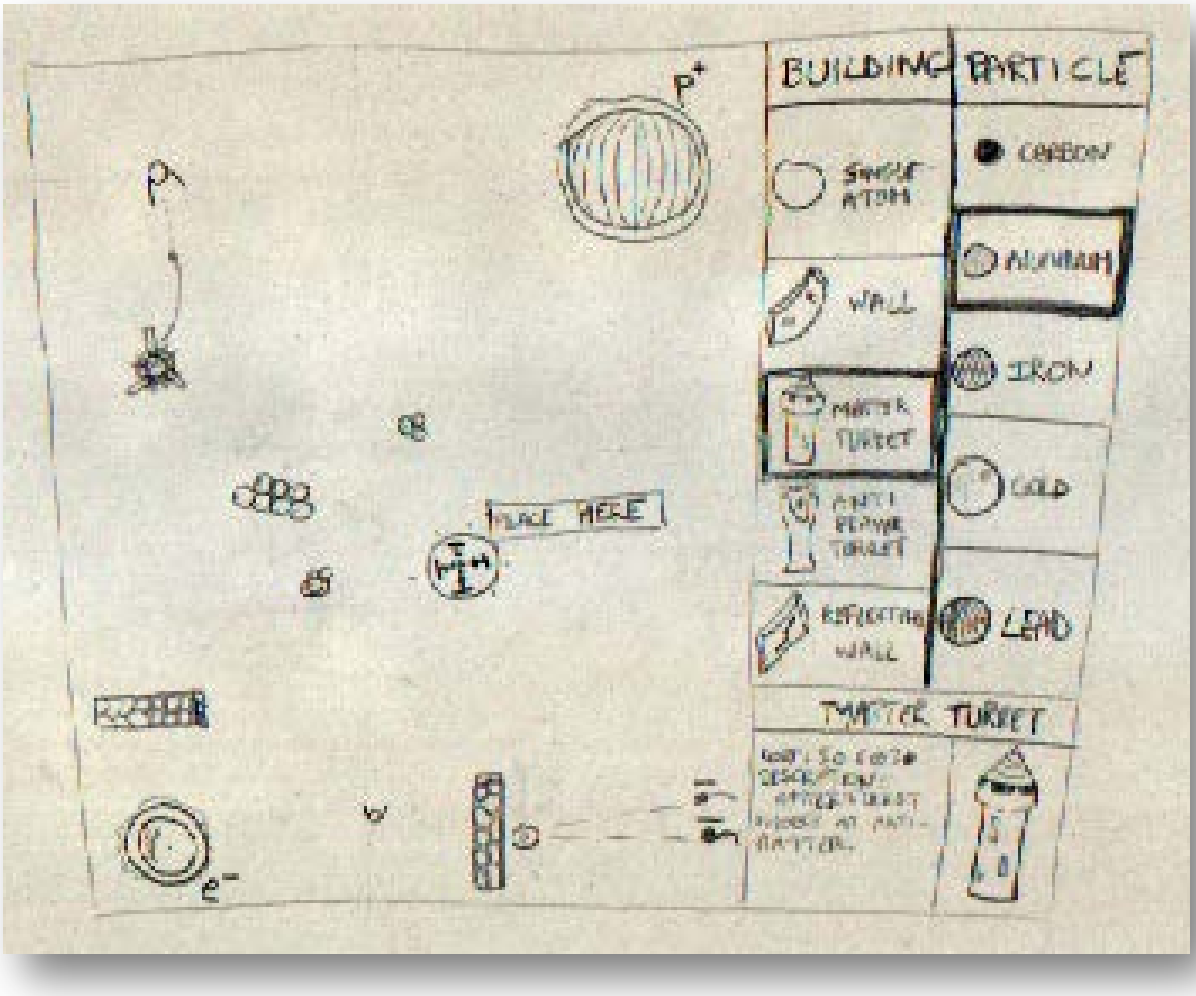


Fig 4: Building screen

DEVELOPMENT SCHEDULE

FUNCTIONAL MINIMUM

At this stage, the player will be able to move on the map, craft atoms and build structures.

- Environment setup (camera, basic rendering…)

✓ Building screen and game screen

✓ Basic HUD for atom crafting

✓ Basic building framework

LOW TARGET

At this stage, we can have several players on the map, but they cannot really interact with each other.

✓ Collision detection for players

✓ Basic map with sources

✓ Basic menus (starting screen, game options…)

✓ Multiplayer gameplay

DESIRED TARGET

At this stage, the gameplay follows the scenario and you can actually play the game in its whole.

✓ Antimatter waves and dark architect inclusion

✓ Better rendering (textures, lights…)

✓ Character modeling

✓ Interactions between players (atoms effects, stealing, turrets, …)

✓ Map design

- ✓ Advanced menus

HIGH TARGET

These are features that will make the game nicer to play.

- ✓ Visual effects for the atoms
- ✓ Physical building simulation using structural analysis
- ✓ Sound effects
- ✓ Animations for the story

EXTRAS

- ✓ Destruction using rigid-body simulation
- ✓ Additional physical effects (earthquake)
- ✓ Character animation
- ✓ Advanced maps

SCHEDULE

A: All, MP: Mattis, MZ: Marco, R: Romain

		6.3	13.3	20.3	27.3	3.4	10.4	17.4	24.4	1.5	8.5	15.5	22.5	29.5
Game Proposal	3h	A	A											
Prototype chapter	8h		A	A										
Environment setup	8h		R	R										
Building and game screens	10h		MZ+ R	MZ+ R										
Atoms crafting HUD	6h		MZ	MZ										
Basic building framework	12h		MP	MP										
Collision detection	10h				MZ	MZ								
Basic menus	10h				R	R								
Basic map	8h				MP	MP								
Multiplayer gameplay	6h					R								
Interim report	6h							A						
Antimatter	16h					A	A	A						
Better rendering	14h					MZ	MZ	MZ	MZ					
Character modeling	6h						R	R	R					
Interaction between players	10h						MP+ R	MP+ R	MP					
Map design	4h						MP	MP						
Advanced menus	6h						R	R	R					
Alpha release	20h								A	A	A			
Playtest	15h										A	A	A	
Conclusion and presentation	20h											A	A	A

ASSESSMENT

WHO SHOULD PLAY?

Our game is designed for everyone who likes to:

- Be challenged: our goal is to make the game decisions really challenging, so that in order to win you will have to find a fine balance between common and individual objectives.
- Kick asses: by using cleverly the different atoms properties you can annoy your opponents and destroy their defenses.
- Be crazy: use as many structures as you want to build your defense.

WHY SHOULD YOU PLAY?

The main strength of our game relies on the fact that each player has his unique style. To get the upper hand, you have to constantly adapt your gameplay and try to find the weaknesses of the other players.

Besides this game is fun in two different ways. First, you can build gigantic and crazy atoms structures. Your only limit is your imagination. Second, you can destroy other player's constructions. This multilateral competition will put a great strain on your ability to make the good decisions.

PROTOTYPE CHAPTER

PHYSICAL PROTOTYPE

BASIC SETUP

For the physical prototype of our game, we chose to make our game turn-based. Every round, players are provided new electrons and nucleons with which can choose to craft atoms, build structures. Alternatively, they can steal from other structures or use one of the interaction atoms (e.g. throw Uranium).

We play the game in 2 player mode and one of us plays the "bank" (providing new nucleons and keeping score) and the Dark Architect. The game is played on a hexagonal grid with small playing figures.

Fig 5. A fun game, for all ages.

PLAYING RULES

Each player gets some amount of nucleons every round. He can first craft atoms ("buying atoms from the bank") and then move and place them. Movement is restricted to 10 steps on the grid (and you may use less.).

A player can buy Carbon for 10, Uranium for 30 and Turrets for 60 nucleons. You can craft as much as you like, as long as you have enough resources. Instead of building, you can also steal from your opponent by moving close to an atom in his camp.

With Carbon you can build your defense: atoms protect the closest source from damage. You can throw Uranium from your position, causing every atom in a radius of 5 to be destroyed. Turrets are used to reduce by one the actual number of anti-matter particle aiming at the related source.

At the beginning of each round, you are given 20 new nucleons from the main source, to which are added 10 per carbon and 20 per turret that protects the center. Cooperation is very well rewarded, but as we may not stress enough: don't be a fool…

Fig. 6. The board filled with atoms and turrets.

After each round, the Dark Architect rolls two dice to determine the damage dealt to the structure. The minimum of the two numbers is the amount of atoms that each individual player loses from the protection of its own source; the maximum minus the minimum is the amount of damage that the central source will absorb. We multiply the outcome by 2, 3… over time, to make the rounds harder and harder.

PLAYING EXPERIENCE

The rules mentioned in the previous paragraph are the result of a long tuning process: we started to play with almost no rules and eventually added new ones as soon as we hit a point where they were needed. The early versions were started over quite a few times.

We tried a set of rules that induced a longer lasting game, to get a sense of the long term experience. In this version for example, another die was rolled to determine after which round the antimatter would come. It resulted in better preparation and strategic planning from the player, which gave a pleasant touch to the game.

For the presentation, we tuned our parameters to give the game a quicker. We felt that showing all the essential effects and possibilities was more important than gameplay or resemblance to the final product.

We found out to be potentially better not to use the real-world numbers given by chemistry to craft atoms, as it is easier for players to remember simple numbers (like 10 for Carbon). Also we may consider dropping electrons out of the building process by giving player and infinite number of them. This would also give us one parameter less to tune and make it easier for the players to craft, having but one number to care about.

We realized that we were going to need a lot of time to tune our parameters (costs, damages, etc.), but we believe it will work out. The board game is undoubtedly different from the actual one, but we were already able to observe some dynamics that we hope to see in our final programmed version.

INTERIM CHAPTER

DEVELOPMENT STATE

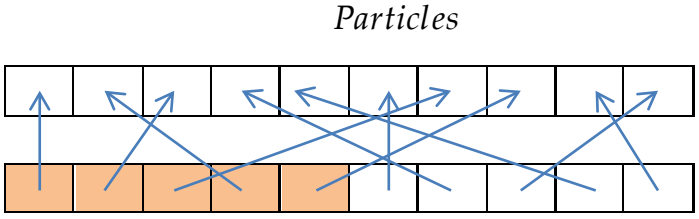
PROGRESS

Our game is progressing well. We finished the low target and are already well advanced inside the desirable target. For the moment, we are on time on the schedule, but we are careful because we think that testing the game and tuning the parameters might require a lot of time.

Some tasks have proved harder than expected. However, the time we spent thinking about software design definitely helped. Furthermore, using libraries in particular for the physics engine, allowed us to stay focus on the game itself.

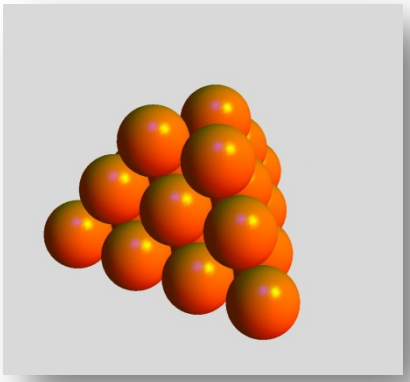
BUILDING FRAMEWORK (MATTIS)

Because we needed to be able to add and remove a lot of particles during the game, we designed a specific data structure that allowed insertion and deletion in constant time, without memory allocation. We used a preallocated array of particles, that acts as memory addresses and an array of integers, which points to particles that are to be drawn.



Keeping at all times a full mapping between indices and particles, defining as active the first k indices and using smart swapping for insertion and deletion, we get our required operations in constant time.

On top of this memory structure, we added a canvas for crystalline structures that determines the position of construction atoms. We used the cubic face centered representation, because it is fairly common and corresponds to a 3D rectangular grid of atoms whose even layers would have been translated half a unit forward and half a unit downward.



Obviously, it would have been impractical to store all possible atom locations; we therefore had to figure out a formula that transformed the 3D coordinates into crystal ones.

PHYSICS ENGINE (MARCO)

We tried out several physics engines and evaluated their performance and ease of use. Specifically, our requirements were stability and performance with a high body count. We are now using Jitter, a very lightweight open-source C# library that implements the famous rigid body paper of Gundelman et al. It's not specifically implemented for XNA, but the guy behind this engine uses XNA himself and therefore the code is well optimized with regard to our enemy number 1: the .NET garbage collector.

The integration was not too hard as the library is easy to use. We used customized collision callbacks to integrate parts of our game logic. Here I lost some time, because the library is completely undocumented. We still have some problems because we add and remove bodies from the physics system a lot, which is not what the developer optimized for.

CHARACTER MODELING & ANIMATION (ROMAIN)

We decided to have nice animated character models for the players and for the Dark Architect instead of ugly colored spheres. In order to do this, we used Maya to design the animated models, and the SkinnedModelSample available on AppHub to import it into XNA.

It took me a while to get familiar with Maya's interface and to practice with several tutorials (modeling, rigging, animating…). Even if I am still a beginner, I feel like I can already do cool things.

To model the character I started from scratch and created the shape I had in mind. It was the easy (or should I say not so hard) part. Then I tried to animate it using a simple skeleton. But after several tests I realized it was too tedious even with inverse kinematics handles. I discovered that I could make this task easier by designing rigging controls. I learned this nice technique from online tutorials and I finally managed to create useful rigging controls on my mesh.

The second part, I am currently working on, is to animate this model. Now that I have my rigging controls I can create poses quite easily. The complicated task is to create an animation which looks realistic. I spend a lot of time studying other games' animations.

Finally the last part is to integrate these animations into our XNA game. We used the SkinnedModelSample as a starting point and tuned it to fit our needs. We tested a simple animation of our mesh and it works. So now, I need to create the different animations (walking, throwing atoms, stealing, sleeping…).

OTHER DETAILS

Dealing with a lot of objects the transferring of vertices to the GPU is quickly becoming the bottleneck, so we decided to implement a basic instancing technique which only transfers the geometry once to the graphics card and then only transfers the transformation matrices. We gain about 15 frames per second with this option activated.

We implemented support for sound handling, using the XACT editor and format. The XNA API in itself did not take long to figure out but we still need to work on sound choice and post-processing.

SCREENSHOTS

MENU

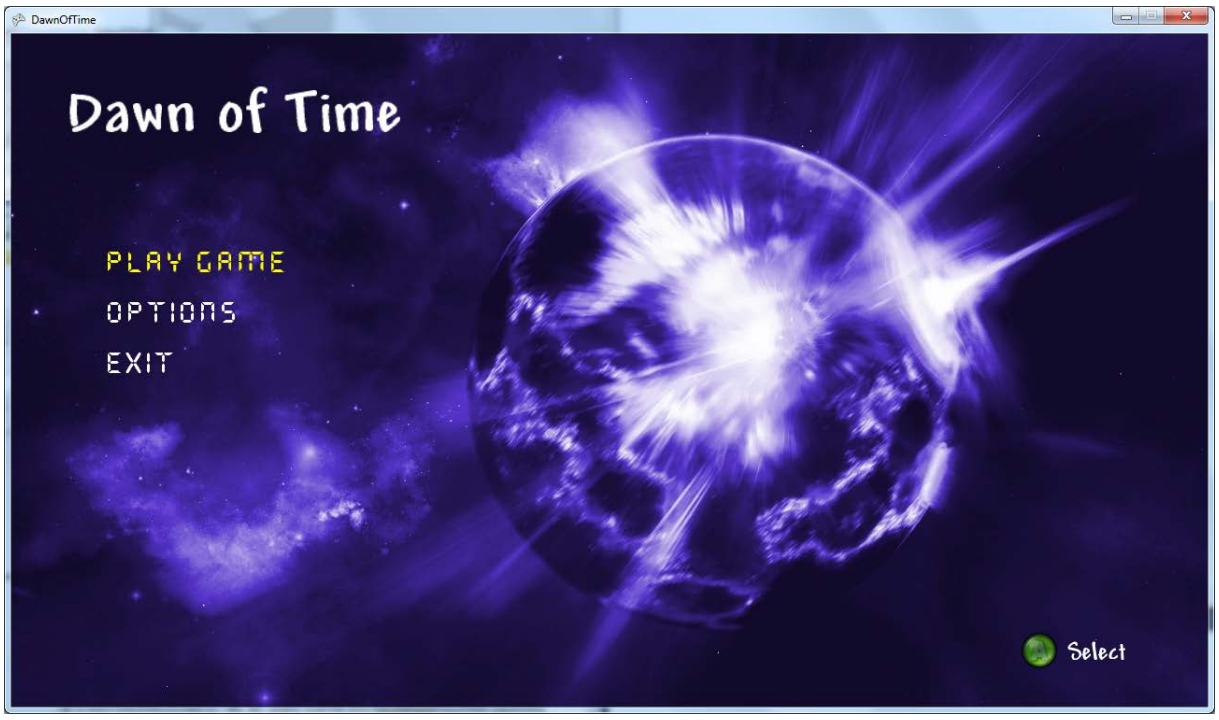


Fig : Start menu

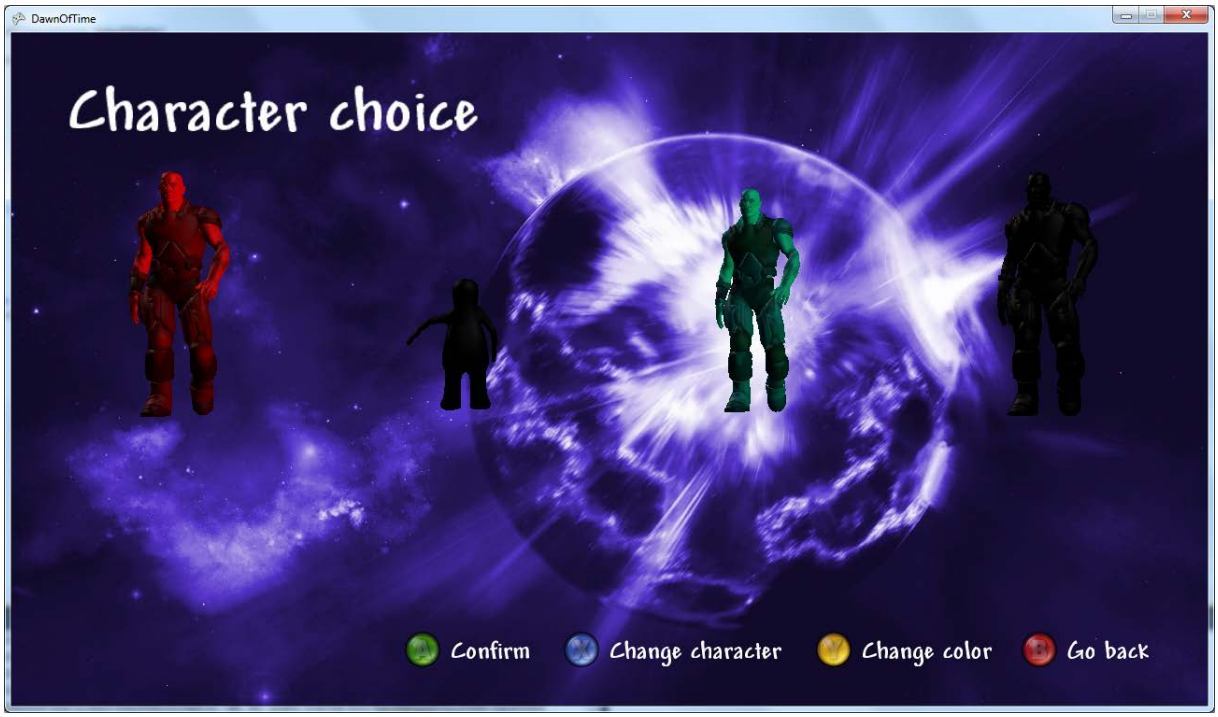


Fig : Character model choice

GAMEPLAY

Fig : We can play up to 4 players

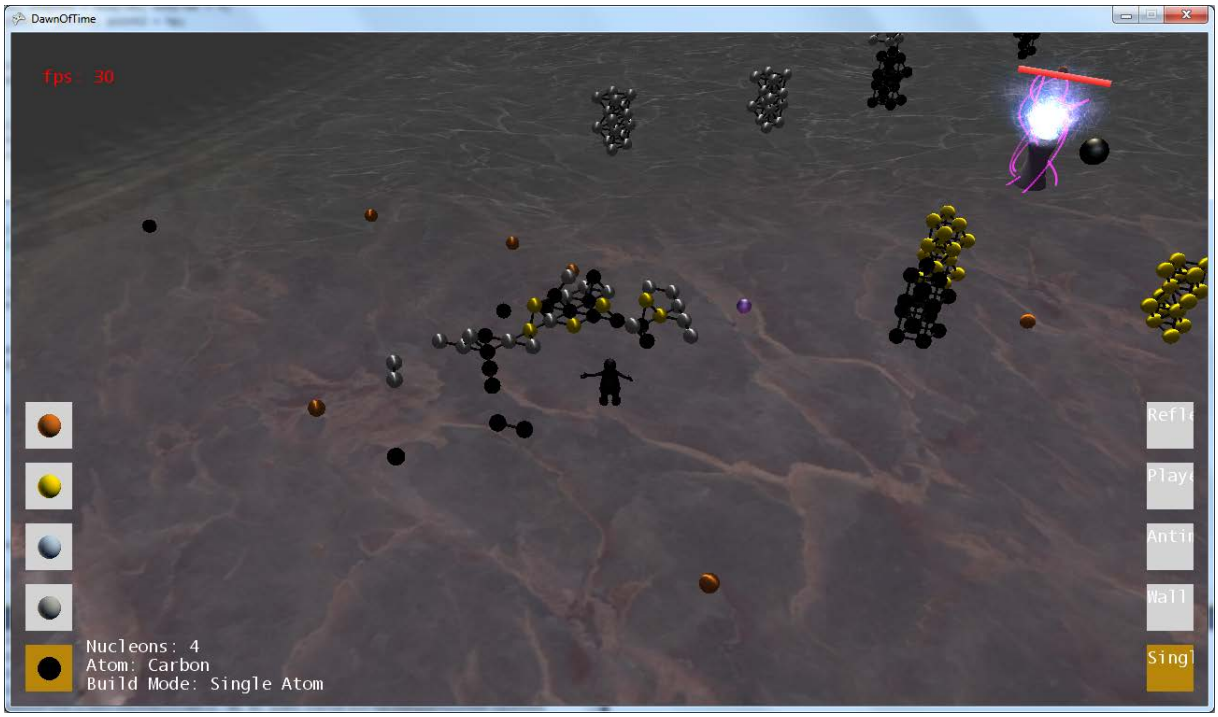


Fig : Building mode, you can easily create structures around your position

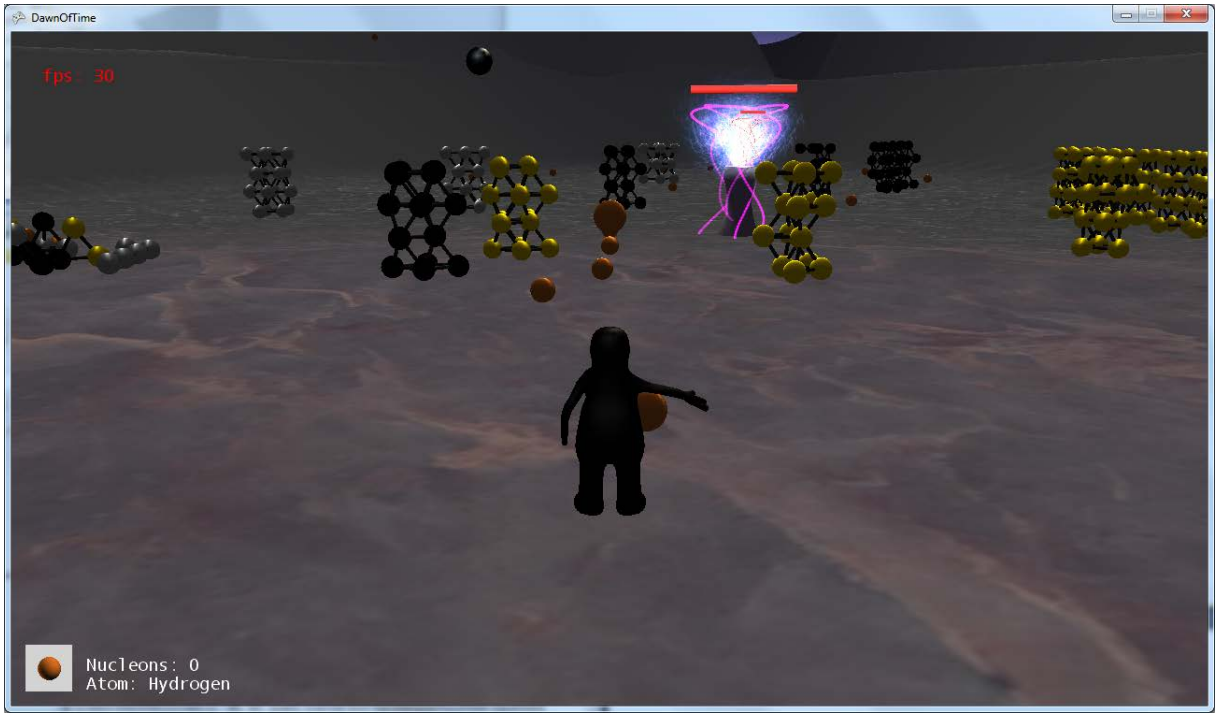


Fig : Game mode, you can move on the map, throw atoms, steal atoms…

GAME DESIGN

GAMEPLAY ADJUSTMENTS

We slightly modified our gameplay since the formal proposal, for reasons spanning from ease of programming to fun improvement.

We decided to remove atom crafting to make the game more fast-paced. Instead, the player directly selects the building or atom he wants to create and the corresponding number of nucleons is subtracted from the player's purse.

To allow for a bit of strategic planning and improve the ability of players to annoy each other, we intend to have a pseudo-realistic AI for the Dark Architect, that would randomly walk around the map throwing waves whenever it sees fit, allowing players to build mazes to redirect it to others. We also plan to have it respond to Hydrogen and Xenon thrown at it. We are aware it will add some tuning work for us but are also confident it will make things more interesting and challenging.

SCHEDULE UPDATE

Target	Task	Scheduled week	Progress	Comments
FM	Environment setup	20.3	DONE	
FM	Building and game screens	20.3	DONE	
FM	Atoms crafting HUD	20.3	DONE	
FM	Basic building framework	20.3	DONE	
LT	Collision detection	3.4	DONE	
LT	Basic menus	3.4	DONE	
LT	Basic map	3.4	DONE	
LT	Multiplayer gameplay	3.4	DONE	

Target	Task	Scheduled week	Progress	Comments
DT	**Antimatter**	17.4	DONE	
DT	**Better rendering**	24.4	IN PROGRESS	Particle system : MOSTLY DONE Lighting : TO IMPROVE
DT	**Character modeling**	24.4	IN PROGRESS	Modeling in Maya : DONE Rigging in Maya : DONE Several model variants : TO DO
DT	**Atoms behavior**	24.4	IN PROGRESS	Integration into physics engine : DONE Action callbacks : TO DO
DT	**Map design**	17.4	DONE	
DT	**Advanced menus**	24.4	IN PROGRESS	Design : DONE Character choice menu : TO DO Winning screen : IN PROGRESS
	Alpha release	8.5		
	Playtest	22.5		
	Conclusion and presentation	29.5		
NEW TASKS				
DT	**Better HUD**	24.4	IN PROGRESS	We are not fully satisfied of our HUD and we would like to make it look nicer.
HT	**Background music & Sound effects**	1.5	IN PROGRESS	Sound integration : DONE Music choice : IN PROGRESS Sound effects matching : TO DO
HT	**Character animation**	1.5	IN PROGRESS	Animation framework in XNA : DONE Animation test : DONE Animations in Maya : IN PROGESS
HT	**Dark architect AI**	1.5	TO DO	Basic walk avoiding buildings Basic behavior choices
HT	**Even better rendering**	1.5	TO DO	Shadows Character texturing

ALPHA CHAPTER

OVERVIEW

Since the interim release, we have made quite a few adjustments to our game, starting by optimizing rendering to allow for a smooth gameplay. We completed the Dark-Architect AI, and now have our fully operational turrets. All garbage collection issues were solved and the game is now entirely playable.

IMPROVING PERFORMANCE

GARBAGE COLLECTION

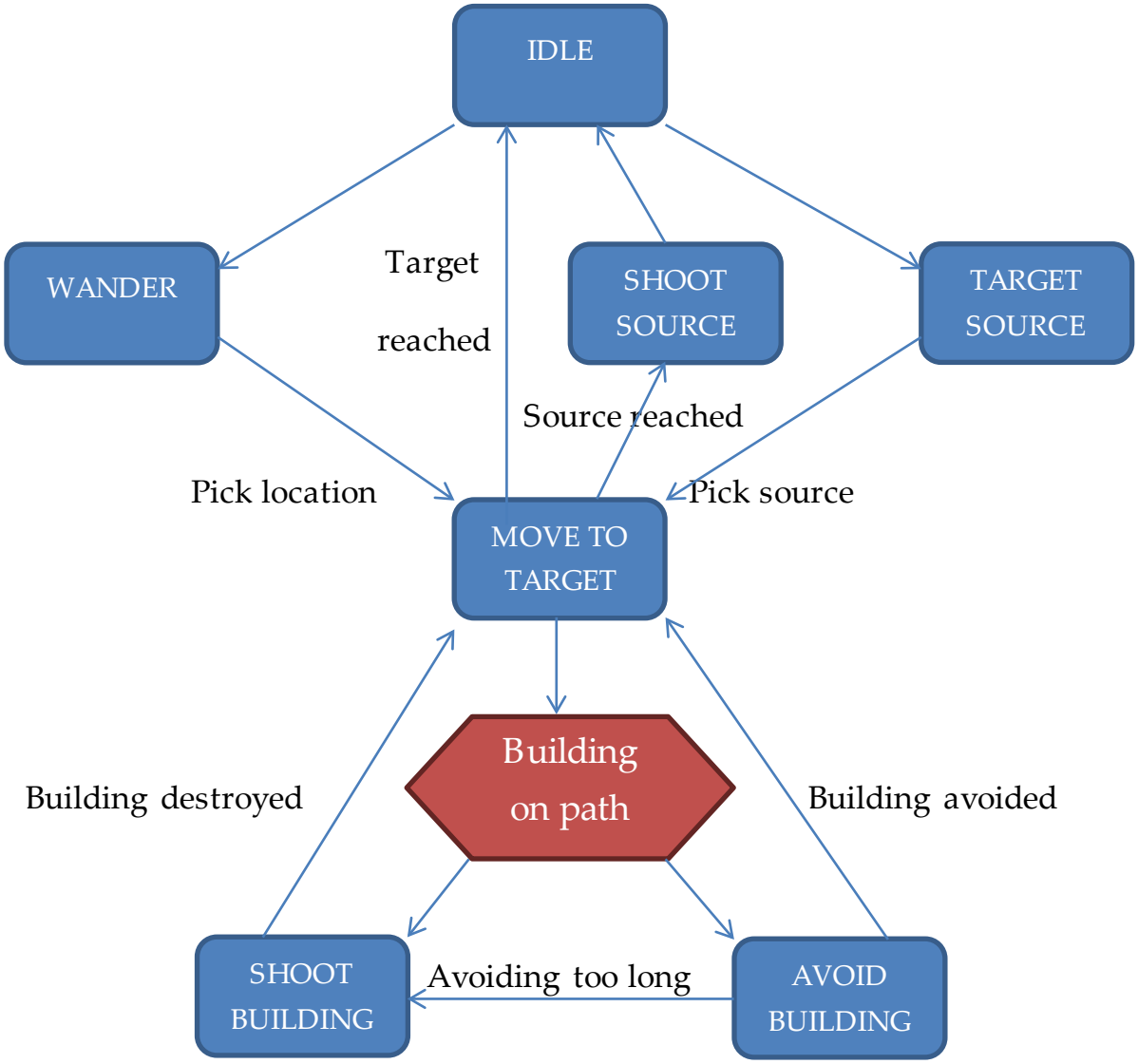
To fix our garbage collection issues, we had to modify our physics engine library, which used to run on generic collections and enumerators, known to induce allocations. After small restructuration of the sources, we got to a point where garbage collection would happen every ten minutes or so, which we deemed satisfactory.

RENDERING

The two main rendered objects are spheres (for particles) and cylinders (for links). Observing that translating a vector is far more efficient to do than a general matrix multiply and using the rotation invariance of the spheres, we were able to simplify the computations in the shader, by shifting them to the CPU and factorizing them. Using the fact that links only appear in structures and therefore have a fixed number of possible orientations, we were also able to reduce the GPU's workload for cylinders. In the end, we observe a small decrease of the frame rate after 200 particles, and estimate that the maximum number of particles in a game will not have it go below 30 fps.

ARTIFICIAL INTELLIGENCE

We implemented a state-machine AI for our main foe, for a better human-like behavior. The diagram is the following:



All parameters (transition probabilities, strength of the waves, speed…) are functions of the game time, in order to make the game more and more challenging and be sure it ends.

TURRETS

We decided to slightly alter the turret's behavior we initially had in mind. Reflectors are now turrets that apply a reflecting force field in a nearby perimeter to antimatter particles. Anti-player turrets shoot at players and "kill" them if they stay around too long, causing the player to respawn near his source. To prevent rendering glitches, turrets cannot be built in a specific radius around other turrets, sources, Architects and structure atoms.

MISCELLANEOUS

To make antimatter waves more effective (and fit more in the theme), antiparticles are now attracted to the sources they are close to. Also, waves are now directional as the Dark Architect sends particles with an angle of 90° in front of him.

We decided to alter the players' movement to ease animation and justify our physically-based movement: Architects are now jinn-like creatures that levitate over the map when moving.

SCREENSHOTS

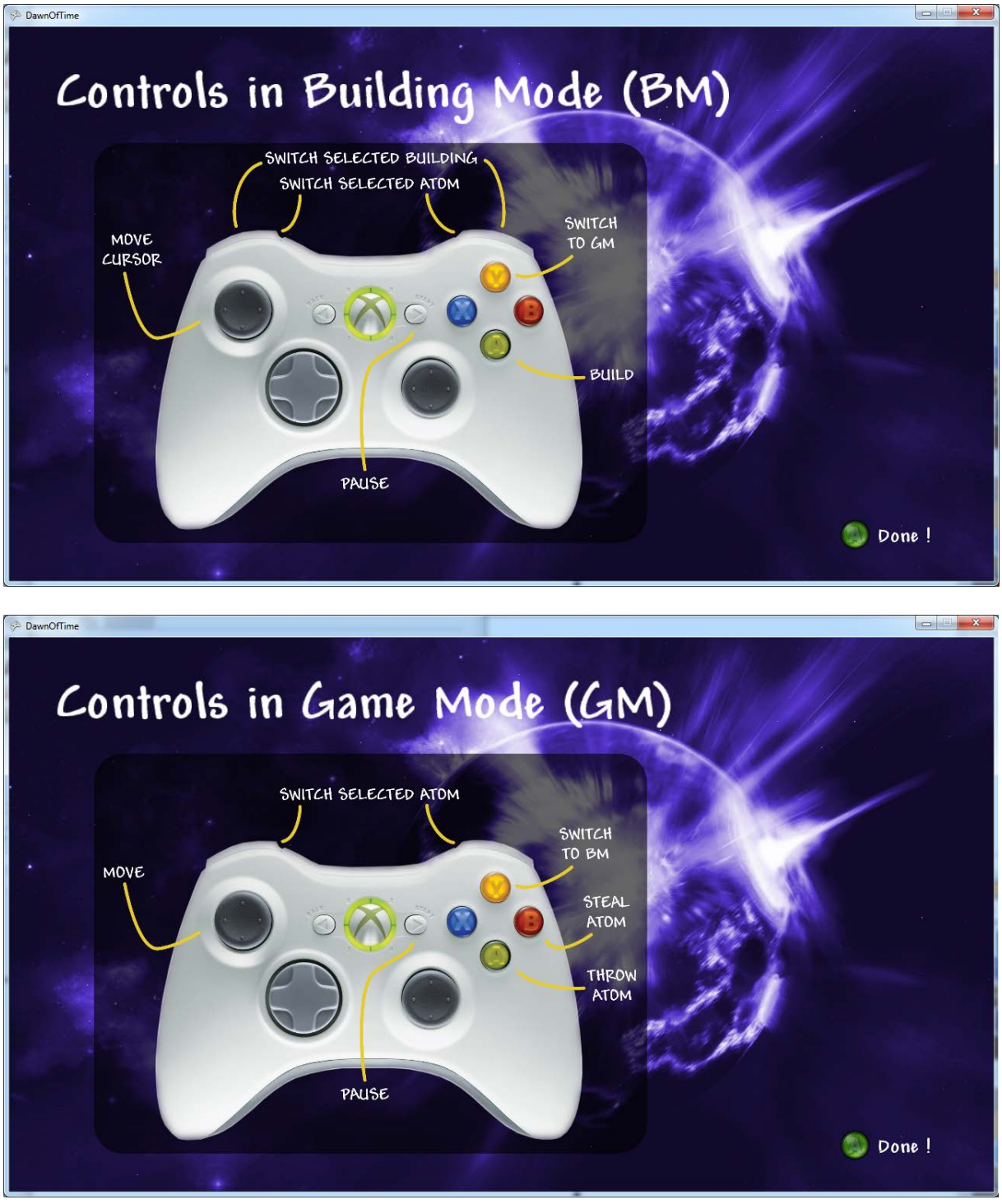


Fig: Help screens with the controls.

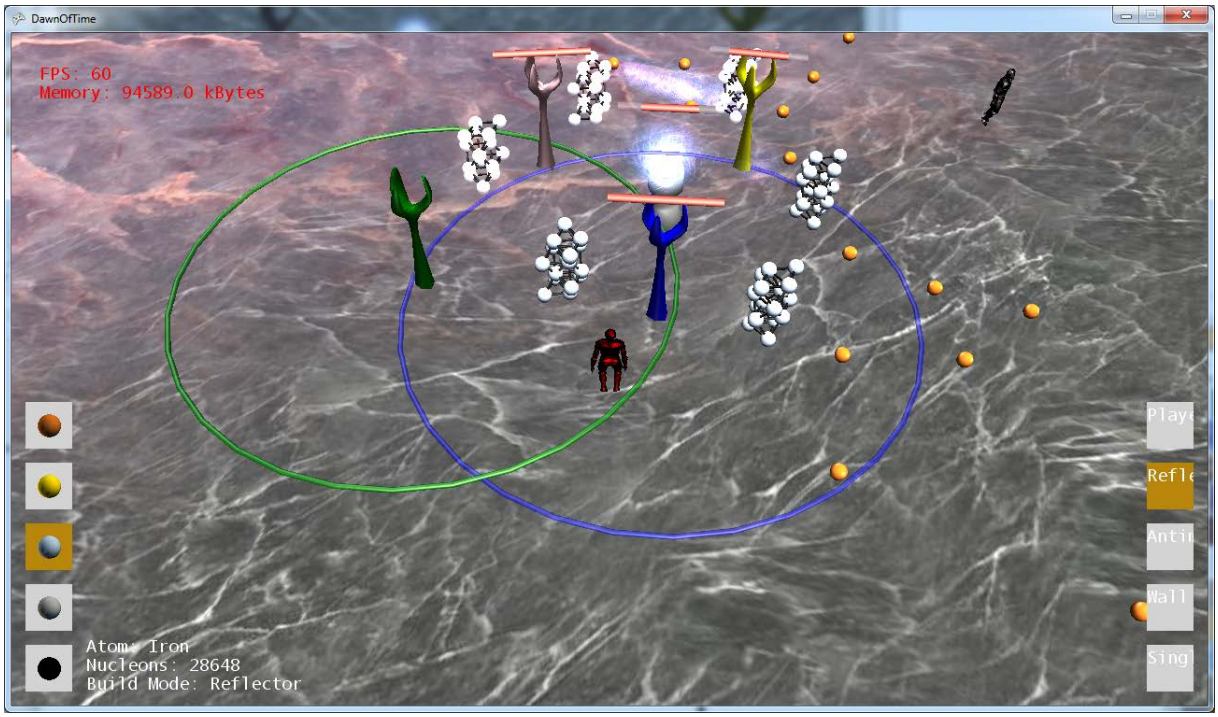


Fig: Building mode with some walls and turrets, and the dark architect having just thrown antimatter.

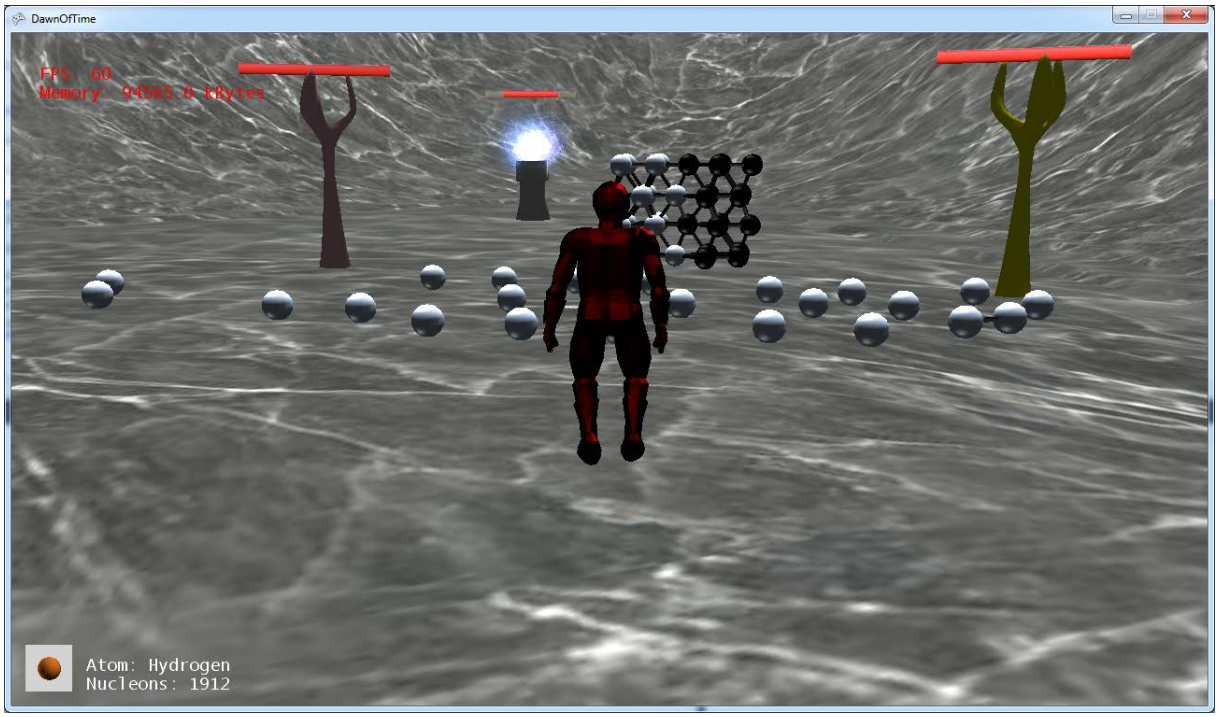


Fig: Game mode: the player is moving toward his defenses.

SCHEDULE UPDATE

Target	Task	Scheduled week	Progress	Comments
FM	Environment setup	20.3	DONE	
FM	Building and game screens	20.3	DONE	
FM	Atoms crafting HUD	20.3	DONE	
FM	Basic building framework	20.3	DONE	
LT	Collision detection	3.4	DONE	
LT	Basic menus	3.4	DONE	
LT	Basic map	3.4	DONE	
LT	Multiplayer gameplay	3.4	DONE	
DT	Antimatter	17.4	DONE	
DT	Better rendering	15.5	IN PROGRESS	Particle system : MOSTLY DONE Lighting : TO IMPROVE
DT	Character modeling	24.4	DONE	Modeling in Maya : DONE Rigging in Maya : DONE
DT	Atoms behavior	24.4	DONE	Integration into physics engine : DONE Action callbacks : DONE
DT	Map design	17.4	DONE	
DT	Advanced menus	24.4	DONE	Design : DONE Character choice menu : DONE Winning screen : DONE
DT	Better HUD	22.5	IN PROGRESS	We are not fully satisfied of our HUD and we would like to make it look nicer.
HT	Background music & Sound effects	22.5	IN PROGRESS	Sound integration : DONE Music choice : IN PROGRESS Sound effects matching : TO DO
HT	Character animation	22.5	IN PROGRESS	Animation framework in XNA : DONE Animation test : DONE Animations in Maya : IN PROGESS
HT	Dark architect AI	1.5	DONE	Basic walk avoiding buildings Basic behavior choices
HT	Even better rendering	22.5	IN PROGRESS	Shadows Character texturing
	Alpha release	8.5		
	Playtest	22.5		
	Conclusion and presentation	29.5		

Target	Task	Scheduled week	Progress	Comments
NEW TASKS				
DT	**Garbage collection issue**	8.5	DONE	
DT	**Rendering improvement**	8.5	DONE	
DT	**Gameplay integration**	8.5	DONE	Turrets, atoms effects, building restrictions, etc…

PLAYTESTING CHAPTER

OVERVIEW

For our playtesting, it was hard to find a time slot to fit everybody's schedule, so we decided to do it in two steps. For the first round we used the alpha release and tested it with 8 people. Then we used their comments to improve our game. Finally we had a second playtesting round with the improved alpha release (let's call it beta release…) where we managed to get 12 testers (including 4 people that had already played during the first session).

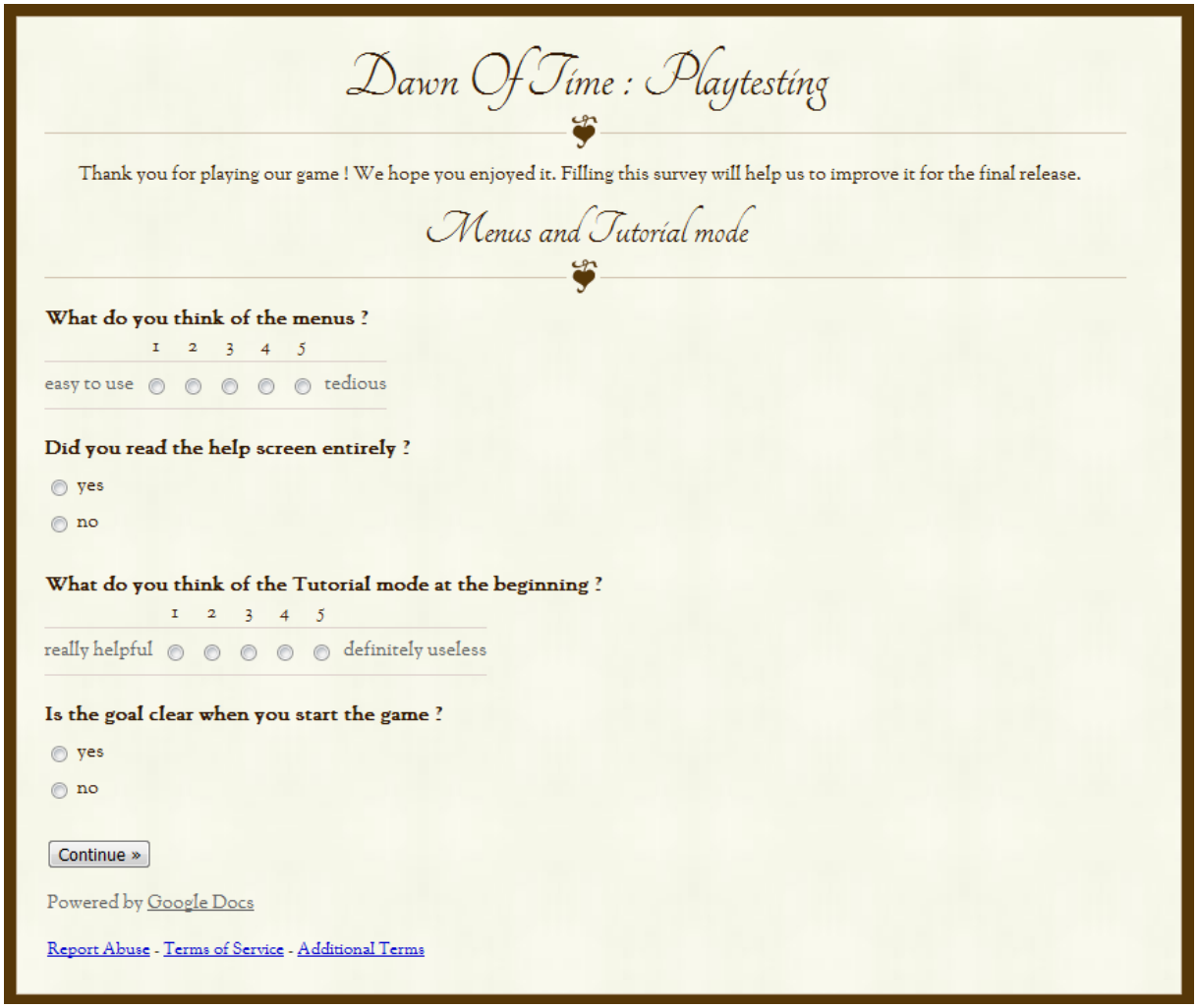
TESTERS

The testers were selected among family members, friends, and random ETH students. The youngest was 16 and the oldest 53 years old. The rate of female testers was around 20%.

People played between 20 minutes up to more than 2 hours, which was for us a good sign that our game could be addictive, thanks to the challenge it proposed. After playing, they filled in a survey to evaluate various parts of our game.

FEEDBACK SURVEY

We used a Google Docs Form to gather the testers' impressions. We had different type of questions to better target the feedbacks we needed. We also gathered their in-game reactions, as well as orally debriefed them afterwards.



Dawn Of Time : Playtesting

Thank you for playing our game ! We hope you enjoyed it. Filling this survey will help us to improve it for the final release.

Menus and Tutorial mode

What do you think of the menus ?

1 2 3 4 5

easy to use ○ ○ ○ ○ ○ tedious

Did you read the help screen entirely ?

○ yes

○ no

What do you think of the Tutorial mode at the beginning ?

1 2 3 4 5

really helpful ○ ○ ○ ○ ○ definitely useless

Is the goal clear when you start the game ?

○ yes

○ no

Continue »

Powered by Google Docs

Report Abuse - Terms of Service - Additional Terms

Fig: First part of the survey.

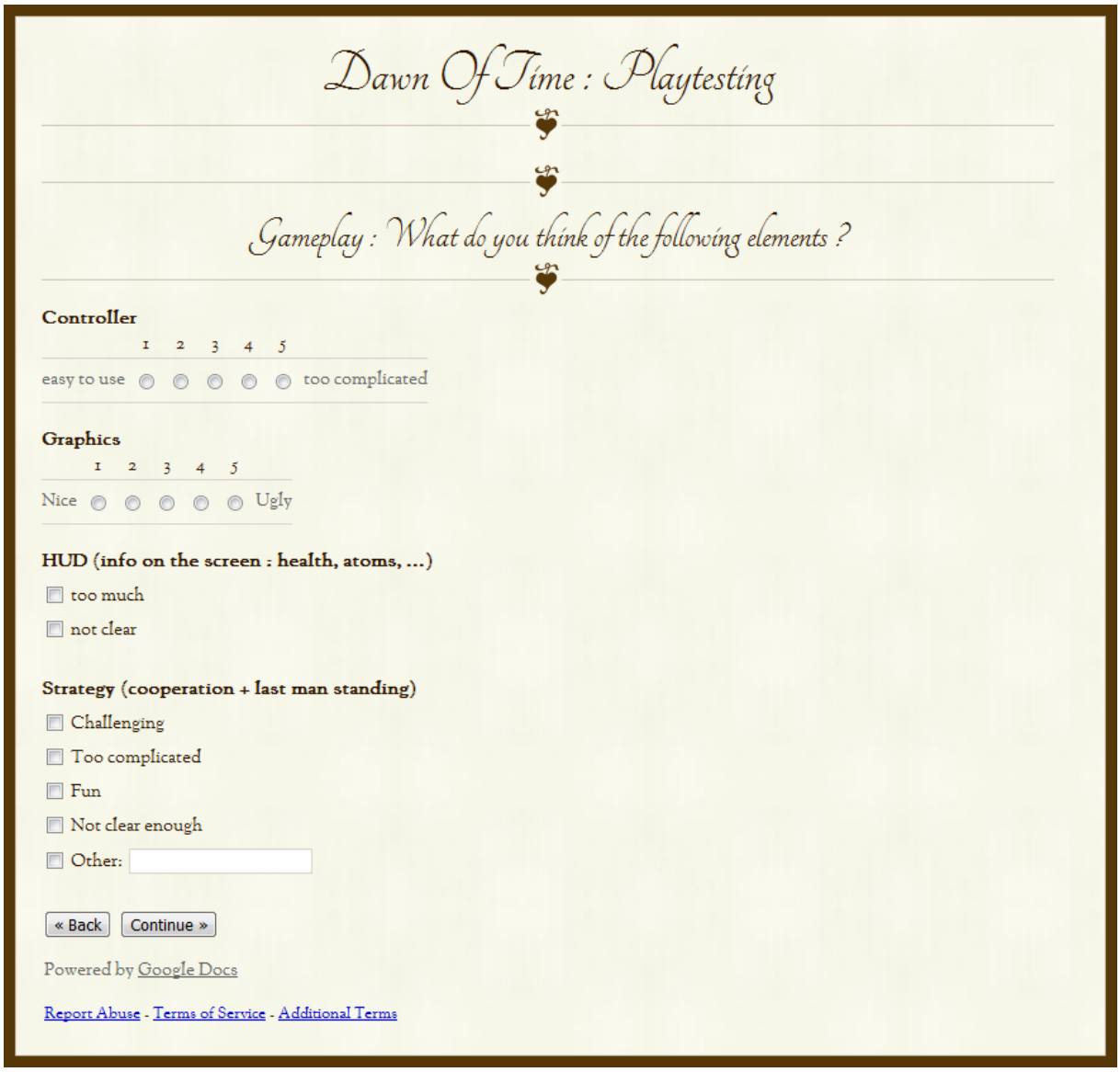


Fig: Second part of the survey.

Fig: Third part of the survey.

ALPHA RELEASE FEEDBACK

We here underline some of the issues that were shown to us and how we solved them.

MENUS

"The number of players should not be hidden in the options"

We agreed that it was really not easy to choose the number of players from the options menu. So we modified the character selection menu, to allow players to join game right after selecting "Play".

"Loading takes too much time"

We found out that on the Xbox the loading of the game was annoyingly time consuming. Furthermore, the help screen with the controllers and the atoms descriptions was only shown after the loading. To solve this problem we did several things: at first, we unified the loading of the animated characters, so that the model and the animation would only be loaded once and at the very beginning of the game. Then we put the help screen during the loading, for players to be able to read it while waiting. However, the problem was that the loading froze the game, and the players could not switch between the different tabs. We chose to parallelize the process by loading the content on another thread.

"Tutorial mode is helpful but you should give more hints"

When we started the playtesting, the tutorial mode was not fully finished yet. We added new hints to help players in their choices. However, we did not want to have too many either, as the players would then probably not read them.

“If you don’t read the hints, you have no idea what you need to do”

For us, it was really clear what the players need to do. However, after getting a lot of “no” to the question “Is the goal clear when you start the game?” we realized that nowhere in the game did we tell the goal of the game. We decided to add a new help screen tab, containing a small description of what players need to do.

GAMEPLAY

Nobody used platinum

The platinum was initially intended to accelerate the building of walls. However we discovered that –apart from us- nobody used it. We decided to delete this feature, while reducing the time it takes to build a wall.

“It is really difficult to distinguish players and sources”

The map and the players were quite dark, causing a lot of players to experience trouble distinguishing their opponents. Even worse, we had two testers that built structures on the wrong sources, not remembering which one was theirs. To solve this important issue we modified our shader to add colored areas around the players and the sources. We also modified the color of the health bars. Now, each player can directly see where its source is located and distinguish its opponents and the Dark Architect (whose size we also increased for dramatic purposes).

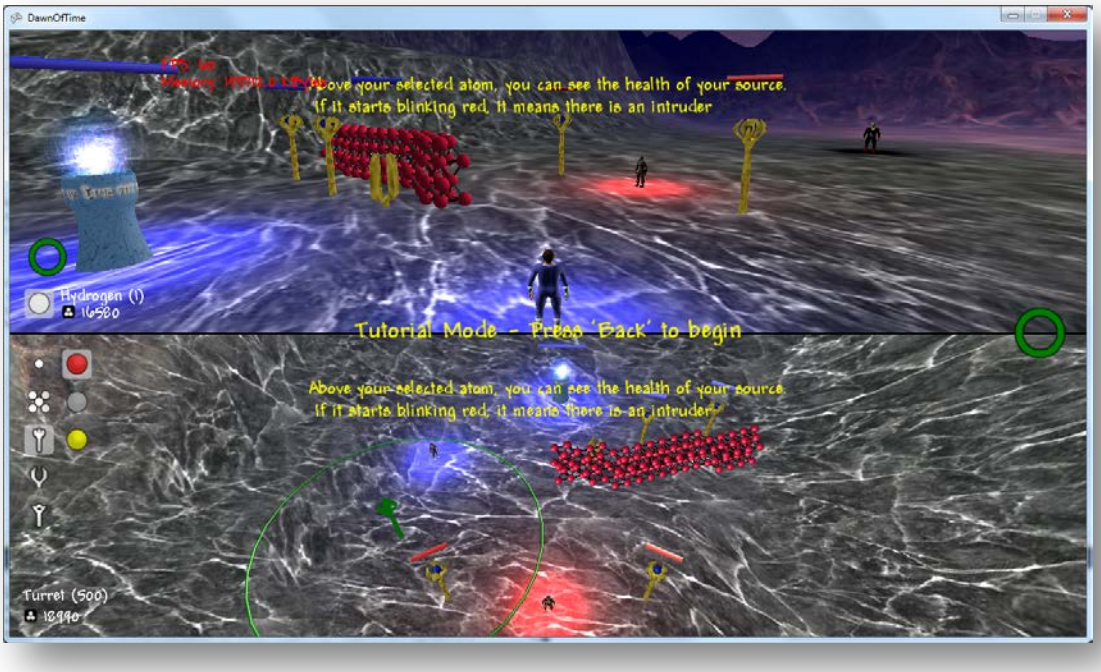


Fig: Color lighted areas under players and sources.

"Controller rumbling would be nice"

Several testers pointed out that rumbles for the controller would be a nice feature. In XNA it is really easy to integrate, so we decided to add it, in particular when you hit antimatter (to discourage protecting your source this way), when you are under xenon (to bother you) and when you throw uranium (when it explodes).

One player can repeatedly use Xenon without the other player having any chance to move

In fact, we did not take this into account and you could basically paralyze players by throwing Xenon at them all the time. To avoid this, we added an immunity period. Xenon now lasts 5 seconds, followed by 5 seconds during which you cannot be hit again.

“We don’t know when our source is under attack”

We actually had something in the HUD to indicate sources losing health, but as it seemed, it was not visible enough. We changed it for a blinking warning signal on the health icon, as well as a sound notification.

BETA RELEASE FEEDBACK

During the second playtesting round, it seems that all the issues listed above were solved since we did not get any complaints. We also noticed that the games became easier to grasp, as testers would be asking less questions about what they could do. From these new feedbacks and also some new ideas we had to improve the game, we made some modifications. This improved version should be more or less our final release.

The main source is rarely under attack

Since players need to cooperate for the main source, this one turned out to be overprotected as the Dark Architect did not attack it efficiently. To solve this we modified two things. First, we increased the attraction power of the main source, so that antimatter would have more chances to hit it. Second, we modified the behavior of the Dark Architect. When targeting the main source, he now turns around it while throwing antimatter. Players would now have to constantly reconstruct around it, which is what we want. We still need to test and tune the corresponding parameters to make it efficient.

“Why can we build so high?”

The height of the building was until now not restricted. The problem was that some beginners were not too careful with their first constructions and ended up building in useless high places. To solve this we now restrict structure heights in building mode.

The rendering could be improved

We added textures to the sources and the characters to make them look more realistic. Despite being everything but artists, we found the result to be somehow pleasant to gaze upon. We also intend to add some post-processing effects adding to the radial blur we already have when a player is hit by Xenon. We also tried to add shadow mapping but, as it was really inefficient on the Xbox and as the resolution limit prevented us to get a high quality result, we decided to drop this feature.

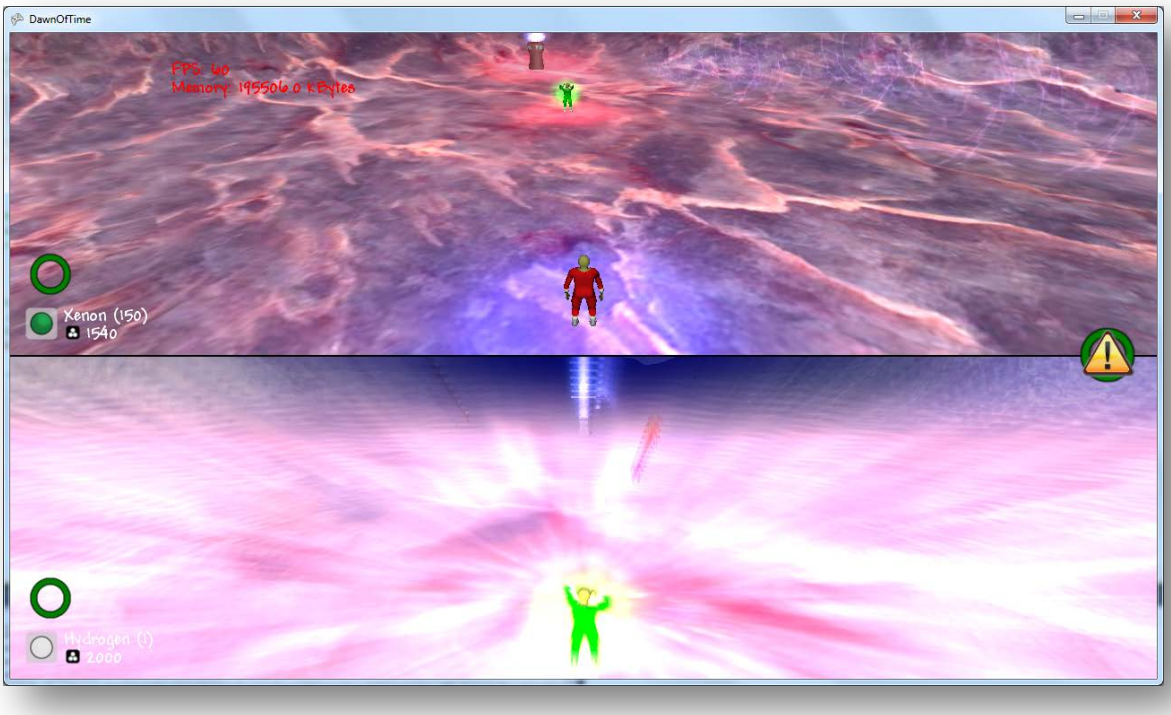


Fig: Xenon effect.

Achievements

To add some challenge to the game and make people want to play it again, we created an achievement section. It is similar to the secondary missions in many flash games. To make it persistent, we integrated a save/load feature to automatically keep the current achievements status.

Fig: Achievement screen

CONCLUSION ABOUT PLAYTESTING

The playtesting session made us aware of many issues of our game that we, its programmers, had not realized. Mostly, interface ease of use and help for beginners were far from achieved. We now think our beta release fixes most of the issues people ran into, and we feel our final version will greatly resemble it, apart from a few possible changes in graphics and AI parameters. The testing session also gave us the opportunity to play our game in length, and experience the fun we had planned on during the proposal, as well as some unsuspected new thrills.

CONCLUSION CHAPTER

OVERVIEW

In 14 weeks of intensive designing and development, we have seen our game go from the state of partial ideas and concepts to a finished playable program. We realized that game-making is a process that draws from a lot of different areas, many of which we had little proficiency with. In the end, we are fully satisfied with our realization, and we still have lots of fun playing it.

Fig: Nothing unusual in the ‘verse…

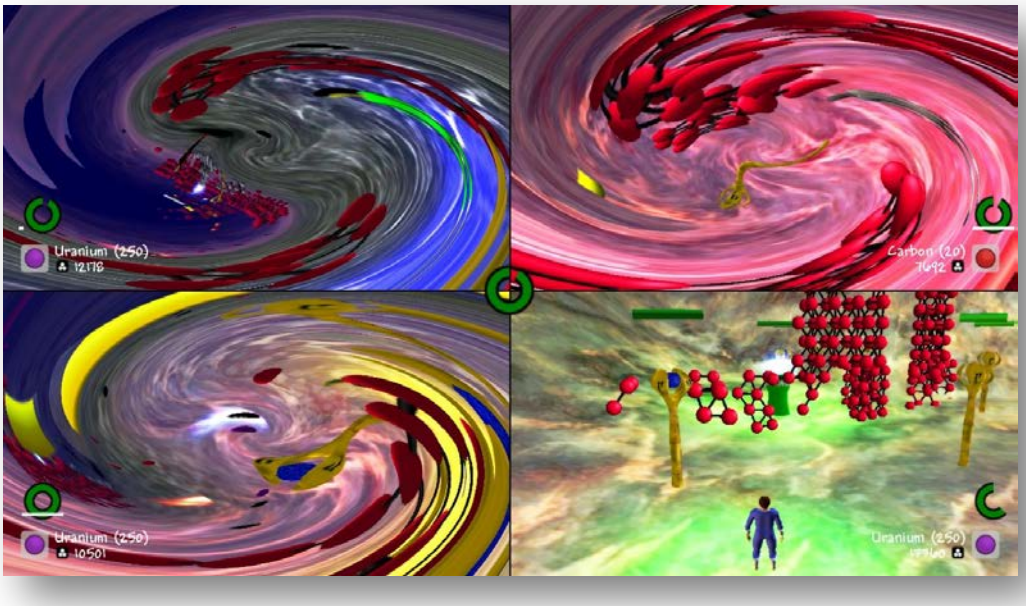


Fig: Twirl effect after being teleported.

REVIEW OF THE INITIAL SCHEDULE

Green features are the ones that have been added during the semester.

Red features are the ones that have been abandoned.

FUNCTIONAL MINIMUM

- ✓ Environment setup (camera, basic rendering…)
- ✓ Building screen and game screen
- ✓ Basic HUD for atom crafting
- ✓ Basic building framework

The functional minimum has been fully completed.

LOW TARGET

- ✓ Collision detection for players
- ✓ Basic map with sources
- ✓ Basic menus (starting screen, game options…)
- ✓ Multiplayer gameplay

The low target has been fully completed.

DESIRED TARGET

- ✓ Antimatter waves and dark architect inclusion
- ✓ Better rendering (textures, lights…)
- ✓ Character modeling

- ✓ Interactions between players (atoms effects, stealing, turrets, …)
- ✓ Gameplay integration
- ✓ Map design
- ✓ Advanced menus
- ✓ Better HUD
- ✓ Garbage collection optimization
- ✓ Controls improvement (simplification, rumbling…)

The desirable target has been fully completed.

HIGH TARGET

- ✓ Visual effects for the atoms
- ✓ Dark architect AI
- ✓ Physical building simulation using structural analysis
- ✓ Sound effects
- ✓ Music and voices
- ✓ Animations for the story
- ✓ Rendering optimization using instancing techniques (for Xbox performances)
- ✓ Tutorial mode

We decided to abandon the idea of the structural analysis since it would have probably required 14 other weeks and would have affected the gameplay in a way we did not intend it to. The animations for the story would have required more time and maybe also more animation skills.

However we compensated with more useful features. In particular we chose to have the Dark architect physically represented in the gameplay instead of just antimatter waves. Thus we had to design an artificial intelligence for his behavior.

Something that we did not take into account in our initial development schedule was the fact that because of the huge structural differences between a computer and an Xbox, performances went way down when we deployed. So we decided to optimize our game and are really proud that our efforts on this matter paid off.

EXTRAS

- ✓ Destruction using rigid-body simulation
- ✓ Destruction using BFS in the building structure
- ✓ Additional physical effects (earthquake)
- ✓ Character animation
- ✓ Advanced maps
- ✓ Achievements
- ✓ Post-processing effects (bloom, twirl, radial blur) to improve players' experience

The more we worked on this project, the more new ideas to improve it we had. In particular, the post-processing effects are features that really immerse the players into the game, while the achievements increase their addiction.

INIDIVIDUAL FEEDBACKS

MATTIS

From my point of view, this course was extremely interesting, in that it allowed us to apply many of the topics we previously learned, such as algorithmic, geometry, shader coding, software development, and so many more, to build a game we would later have fun playing with. Also, it introduced us to a large amount of areas we were not familiar with, or at least had never been taught, such as 3D modeling and animation, multithreading, performance optimization, sound management… Thanks to the XNA framework, and many free libraries we found on the internet, the amount of original code we had to produce was kept to reasonable proportions for the 14 weeks of our team of 3.

One of my greatest regret is that we spent more time making our game run on the Xbox, rather than actually improving its features. Debugging on the Xbox is very tedious, and computers are nowadays much more efficient. Yet, I am glad we actually had to develop a console game, as their profile is very different than the computer ones.

Concerning milestones and schedules, we were relatively on time each one of them (except maybe for the alpha release which lacked a few features we added later on). Yet, we had started way earlier to code – almost right after we had decided towards which game we were evolving-, as we knew it would be a tight schedule.

I have no complaints with the theme, especially because it oriented our imaginations and boosted them, as well as gave us justifications for our game's feature choices.

MARCO

The GPL is basically the biggest course I have ever taken. Compared to other classes at ETH where most of the time you only have to fill the last inner loop of the problem, here you build something from scratch until the end in

more or less production quality. Even though that also means to write lots of boilerplate code, it helps with your understanding in software engineering in general and you really have to work as a team.

I think it helped that nobody in our team was protective about his code, so everybody just changed bits and pieces everywhere and therefore knew a bit about all the code. It was then easier for the discussions (or in the end performance options). From time to time, we did more detailed intern todo-lists and assigned tasks, but when some feature turned out to be bigger, we just handed the tasks over.

I think we were always doing well with regard to the schedule and reached almost all our goals. Some advanced features were maybe done a bit too early and in that way we were quite late with the first version of a really playable game. We should have rooted for a playable game (be it with ugly graphics) a bit earlier. But it was just fun to implement new features!

I really like the XNA framework, also because it's heavily used and you always find solutions to your problems. It's also quite nice that you can compile your code for both platforms. It's a pity that now most laptops are faster than the XBox and the "wow-effect" of being able to program for the Xbox is often overshadowed by the problems.

I'm not such a big fan of the lectures as they are about things that you know already and otherwise quite superficial. It would be really nice if some students of earlier years would talk about XNA and their problems so that you would get input from people that used the XNA libraries and the Xbox for a longer time period.

ROMAIN

The Game Programming Lab has been a really challenging and immersive experience. I learned a lot about the process of creating a game. I did not imagine how useful the initial design could be. In fact, I think we saved a lot of time by putting ideas on paper and really thinking about how they would improve the gameplay.

At the end, I am really happy with the result. The more I play the game, the more I have fun finding new strategies and unlocking achievements. In this sense, I am convinced that we achieved our goal: creating an innovative game with the concept of construction versus destruction and the complex strategy duality. Besides it is really addictive. Contrary to some other games that were presented, our gameplay is quite complex and the playtesting session showed us how difficult it can be for a beginner. But now that we added a tutorial mode and that we simplified some features, the game is playable by anybody. And in a few games, you reach a good level.

Developing for the Xbox has been quite a challenge, because every time we implemented something we had to verify if it was also working on the Xbox. For our game, it was really hard to get good performances since we have a lot to draw and update for each frame. But it was also a huge victory when everything was working fast on the Xbox after dozens of commits…

The XNA framework is nice and gave us the tools to work efficiently since the beginning of the semester. The AppHub samples are also a big help. However, we had quite a lot of garbage collection and compatibility issues. Solving them was sometimes a real pain and took us a lot of time.

Finally I would suggest more feedback about the presentations that we give along the semester. In fact, we wrote a lot of reports and gave a lot of presentations, and I think we could have used more feedback from GPL's staff.

CONCLUSION

So as you can see, we are pretty satisfied with our game. At the beginning of the semester, none of us would have imagined we were capable of such a complete result. Thus, the huge amount of work we put into this project was worth it despite the lack of sleep. The team work has been really profitable since we managed simultaneously to divide the tasks and to decide together on the development. The game we present is not as beautiful as a commercial game; however it is quite finalized regarding the gameplay – at least this is our point of view.

We hope you will enjoy playing our game as much as we do.

MRM GAMELABS