

Computing Project: ATHENA

Kieran Armstrong, Alastair Campbell, Grant Thorburn, Chrisy Carson

May 6, 2014

Contents

1	Group-Based Project Plan	2
1.1	Introduction to Project	2
1.2	Gantt/PERT Chart and Critical Path	2
2	Group-Based Development	2
2.1	Project Planning and Approach	2
2.2	Design of Deliverable	3
2.3	Development of Deliverable	4
2.3.1	Description of Deliverable	4
2.3.2	Deliverable Features	4
2.4	Implementation & Evaluation	5
2.4.1	Implementation Models and Approaches	5
2.4.2	Key areas addressed	5
2.4.3	Evaluation	6
3	Appendices	7

1 Group-Based Project Plan

1.1 Introduction to Project

Our group project, code-named ATHENA, was to create a video game. We discussed some ideas and eventually agreed to create a science-fiction themed game in the RPG genre with a turn-based combat system. We chose this because we thought that it would be an interesting setting and theme for the game and also due to our belief that it would be less complex to implement compared to a more fast-paced action-oriented game.

Once we had an idea of what we wanted to achieve, we approached the project with an approach which was particularly planning-heavy, creating a very detailed project plan with accompanying charts and diagrams, alongside a broad plan of what we intended our game to be like - level design, story elements, characters, etc. We then fleshed out these basic ideas until we had, essentially, the elements of our game in terms of UI design, character sprites, the lines spoken by characters and when they would be spoken, along with a chronology of the story taking place. The game engine of our project was the biggest element of the game and received the most work on it as a result.

We were aware of the scale of our project so we knew we had to keep strictly to schedules. Our project was originally planned out task by task, with an estimate of how long we thought each task would take. Resources were then assigned to the tasks appropriately. Other than human resources, we used several tools (including Microsoft Visual Studio, Microsoft Word, Microsoft Visio, Microsoft Project and Adobe Photoshop) to create or manage our project.

1.2 Gantt/PERT Chart and Critical Path

Please see appendix 10 for our Gantt/PERT chart with critical path.

2 Group-Based Development

2.1 Project Planning and Approach

Once the project plan was created, it was largely ignored. This was probably due to the fact that it was so prescriptive about what must be done and when it must be done that it was overburdening. The project was managed on an ad-hoc basis, with work being where it was needed rather than in accordance with the plan. Due to this, certain tasks fell behind and others received the majority of the work. For example, the game engine got the largest amount of work done on it, whereas sprites, level design, sounds, etc. were nearly completely neglected. I believe that a more lightweight, agile project plan to start with would have benefited the project more than the heavyweight plan we had, allowing us to

attempt to follow the plan more rather than blindly working on whatever suited at the time.

2.2 Design of Deliverable

We adopted a mostly ad-hoc approach to the design of our game. Due to time constraints, it was decided that the time taken to plan the class hierarchy would be better spent implementing the classes themselves. This approach was very successful and led to a very powerful and robust, but complex, game engine. The engine itself was designed to be highly object-oriented and was written in C# on the XNA 4.0 Framework, which turned out to be the biggest flaw in the project. XNA 4.0 was officially dropped by Microsoft and was not supported in the latest versions of Visual Studio, requiring several workarounds to be deployed to be able to use certain workflows. The project was later switched to a free implementation of XNA 4.0 called MonoGame, but this was not without flaws (for example, it was impossible to compile fonts into sprites using MonoGame). However, most elements of our game, such as the story, user interface and sprites were planned and designed prior to implementation.

We quickly adopted a version control system (VCS) for managing our project code and files. Initially, we used `git` to manage our source code and project files. This was later switched to Microsoft's built-in Team Foundation software for better integration with the Visual Studio environment. The Team Foundation Version Control was used to control the source code version while `git` was still used to maintain shared copies of the Project file, UI plans, floorplans, diagrams, etc.

We designed our story in Microsoft OneNote. We wrote some ideas of what we wanted our story to be like, then we wrote a plot overview and finally wrote the story using a storyboard and a set of spoken dialogue for each character.

Similarly, we planned our user interface and character designs in Adobe Photoshop. We sketched out how we wanted our characters to look, then we refined these sketches until we had a complete sprite for each character.

The same method was used for level design. Simple sketches of how the levels should look were created, and then the designs were digitized using Microsoft Visio.

The game engine used a slightly different design method. Rather than planning progressively how we wanted it to be, we had a very strong idea of what would be needed, allowing us to forego the more formal design process used for other game elements. We created a very strong object-oriented structure in Microsoft Visual Studio itself, rather than planning it using a modelling system such as UML first. This approach was chosen due to the fact that it would

take far longer to do in UML first and we had very limited time for this very ambitious project.

2.3 Development of Deliverable

2.3.1 Description of Deliverable

The prototype is mostly comprised of the game engine. Our engine is a very strong class-based inheritance structure featuring objects to draw many types of entity and handles them in a very reliable way, using interfaces to ensure compliance. Our prototype solution also features a test suite to attempt code regressions. To run our prototype, you require a recent version (2010 or later) of Microsoft Visual Studio. To compile the prototype you will need a copy of the latest build of MonoGame (version 3.2 at time of writing)

2.3.2 Deliverable Features

Our prototype is very unfinished. The features that we have implemented are:

- Class-based inheritance structure
- Drawing standalone sprites or sprites out of a spritesheet
- Drawing UI elements (such as buttons) with labels
- Actor (such as player, enemy, etc.) movement with collision detection
- Advanced XML-based level loading system
- XML-based tileset system
- Unfinished system for handling RPG elements, such as experience, health and attributes
- A camera system that tweens to the focused object and allows zoom, rotation, etc.
- Several helper classes which add to the framework (for handling graphics, fonts, etc.)
- Code reflection in game objects (For example, triggers written in C# which trigger when a player walks on them)
- Limited user-interface components - buttons can be displayed, but they are only for show currently.

Features which we have not yet implemented, but plan to:

- Actual gameplay - levels need to be encoded into XML and the unfinished combat system needs finished.

- User Interface - More elements with interactivity (using code reflection like trigger tiles)
- Sounds - These aren't yet implemented but there is some groundwork done in this already such as the ResourceManager helper class.

2.4 Implementation & Evaluation

2.4.1 Implementation Models and Approaches

Our initial plan was to develop our game engine in parts for example, player movement and collision detection. We then realised that some parts relied on other parts being completed, for example Collision Detection needed Level Loader and Player Movement. So our approach changed, we started focusing on individual parts rather than trying to make everything work at once.

Our group utilised an Agile Methodology, agile methodology promotes adaptive planning, iterative development and a project which is able to adapt to change rapidly. this benefitted our project as much of the development was ad-hoc and unstructured. Some flaws with this methodology are that sometimes you make mistakes and time is wasted but the advantages of wasting less time planning development as opposed to actually developing far out weighs the flaws.

Class diagrams are very useful tool during development, we decided not to use class diagrams. We thought that if we never implemented class diagrams that it would reduce development time. this did have the desired effect however if we implemented class diagram the development time may have been reduced. The output however had the same effect the only flaw was some time may have been lost.

At the start of the project we implemented planning first approach, this wasted alot of time as we never followed the plan. We believe that if we utilised an agile methodology from the begining we would could've put the time that was wasted to a better cause.

2.4.2 Key areas addressed

We split our project implementation into five main parts - the game engine, the story script, the graphical design, the sound design and the level design.

Game Engine This involved planning and writing classes to work together as part of our game to support all of the necessary gameplay features. The key features of the Game Engine are:

- Programming Movement Logic - player movement and npc movement
- Programming RPG Elements - this includes - levelling, character stats etc.

- Programming Collision Detection - players interaction with level
- Programming Graphic Handler - this includes Textures, Sprite Sheets etc.
- Programming Story Handler - handles story elements
- Programming Level generator - converts XML documents into levels
- Programming Encounters - triggers battles encounters, triggers story encounters.
- Programming Battle System - this includes damage recieved/taken, item usage etc.
- Programming Sound Handler - this includes background music, battle music and sound effects

Story Script This involved writing the story, dialogue and how the story would progress. during the planning we decided how our story would progress. We also decided on character names and what characteristics they had. We also decided that we wanted our game to have multiple ending depending on user interaction with our game so we thought of 3 endings: good, evil and neutral.

Graphical Design This involved creating graphic assets such as sprites and UI elements. We firstly created sketches of what we wanted our characters and levels to look like, and then we created prototypes in Photoshop, which then became actual game assets.

Sound Design This involved choosing appropriate sounds and background music. We chose appropriate sounds for our game and then we saved these as .mp3 for import to our game.

Level Design This involved designing levels for the game. We sketched how we wanted our levels to look and then we digitized them in Publisher. These levels can then be written in the XML format for our level import tool.

2.4.3 Evaluation

This development will be evaluated by looking at the work done and based on how much of the work was completed. The game engine was the furthest into development. player movement, collision detection and drawing onto the screen were all implemented, the only things that weren't fully operational was the user interface, story elements and sound.

Although the game was never finish we done testing of the engine. We asked a third party to perform gameplay test of the prototype. where collision detection, full player movement, random encounter triggers and test sprites were drawn to the screen. They were also shown the game script and some of the graphic concept.

We also applied unit testing, unit testing is performing test against self contained units such as classes and functions. We used unit test to prevent code regression (i.e change code breaking existing functionality) so when the solution was compiled, if there was any code regression the tests would fail.

Another technique for evaluating software is to use various testing techniques, such as white box/black box testing, top-down and bottom-up testing, etc.

3 Appendices

See appendices:

Appendices

Contents

1	Appendix 1 - Storyboard	2
2	Appendix 2 - Genre	7
3	Appendix 3 - Characters	10
4	Appendix 4 - Levelling	12
5	Appendix 5 - RPG Plan	14
6	Appendix 6 - Items/Experience Plan	16
7	Appendix 7 - Sprites/Graphics	19
7.1	Characters	19
7.1.1	Professor Sinclair	19
7.1.2	Professor's Assistant	19
7.1.3	Military Commander	19
7.1.4	Soldier	19
7.1.5	Elite Soldier	19
7.1.6	Security Guard	19
7.2	Other	19
7.2.1	Battle background	19
7.2.2	Battle UI	20
8	Appendix 8 - Code	21
9	Appendix 9 - Code Documentation	59
10	Appendix 10 - Project File	121

1 Appendix 1 - Storyboard

Storyboard

Wednesday, February 5, 2014

1:15 PM

First Draft of the Script, if you think it needs any changed then make some, make sure to save it as Draft 2 though so we can revise them all before actually finalizing the script.

ACT 1

Scene 1

Fade in from black into a high-tech science lab.

- "Was the implantation a success?", says Sinclair to Dr. Rogers
- "You tell me." says Rogers
- << the game UI boots up and loads >>
- "Good day, Professor. I am your Automated Tactical Heads-Up and Environment Navigation Aid - ATHENA. I will be assisting you in your research into nanotechnology. Connecting to facility security network...
- //"G'Day Professor am your Australian Tactical Heat-up Enviromental Navi Aid - ATHENA.\\
- "Is the AI fully operational?" - Rogers
- "Everything seems to be working better than exp---" - Sinclair
- //"em kinda..." - Sinclair\\
- //"al re-calibrate the system" - Rogers\\
- "ALERT. UNAUTHORISED ACCESS IN SECTOR 7" - ATHENA
- "What the ... get out of here!" - Security
- << A soldier enters the room >>
- "Don't worry, sir! I can take 'em!" - Security
- << security guard shoots a few shots at the soldier and is promptly mauled >>
- << The scientists run in two different directions >>
- <<Sinclair runs into a group of soldiers and is captured >>

Scene 2

Fade in from black into a locked storage closet.

- ATHENA: "Picking up radio chatter... Connecting to secure frequency..."
- //Soldier 1 WAKE UP!-rogers\\
- Soldier 1 : "So, what are our orders sir?"
- Unknown voice : "Would you kindly access the central mainframe. Some security subsystem disconnected it automatically from the central network..."
- ATHENA : "If they gain access to my central mainframe, they will have full access to ARC Research designs and technologies. "
- Sinclair : "ATHENA, can you unlock this door?"
- ATHENA: "Affirmative. Overriding door security... I suggest you find a weapon, Professor."
- <<Player is given control. Door opens.>>

Act 2

Scene 1

The player arrives at the central mainframe room. The door is locked

- //hate forgetting my keys - sinclair\\
- ATHENA: "Attempting to override door security..."

- <<ALARM GOES OFF>>
- ATHENA: "This door uses time-based encryption technology... you will have to hold off enemy forces until I can break it..."
- << TIMER: 30 seconds. >>
- Unknown voice: "He's at the mainframe room. Would you kindly deal with him."
- ATHENA: "I cannot breach this security, Sinclair. You will have to manually override it in the Security room."

Scene 2

The player is inside the security room and overrides the door.

- Unknown voice: "Sinclair... you cannot stop me. You are an insignificant distraction. Would you kindly join me or be destroyed."
- ATHENA: "Detecting anomalous network traffic... automated defenses are coming online... and I can't disable them, Sinclair. Sorry."
- //Damn what use are you?- Sinclair\\
- Sinclair: "Its fine ATHENA, I better head back to the mainframe room anyway."

Scene 3

The player returns to the central mainframe room and his way forward is blocked by the military AT.

CUE MINIBOSS FIGHT with epic music

- Cmdr. Kemp: "Sinclair... you cannot stop us. Your only hope is to cooperate. Help us improve the technology you built."
- Sinclair: "I built the technology to help humans, not to kill them."
- ATHENA: "Commander Kemp, you have endangered humanity. You must be stopped."
- //Stopped you will be - Sinclair\\
- << AWESOME BATTLE SEQUENCE>>
- << Kemp is defeated, lies on the floor dead >>
- //Time to teabag - Sinclair\\

Act 3

Scene 1

The Player enters the hall way leading to the Server room only to find the Cause of all this chaos.

- Unknown Voice: "So you are finally here Professor. I will forget all that has occurred here tonight with the deaths of all my men and what not, if You'd Kindly join my side to rule the world".
- ATHENA: "Sinclair will never join you!".
- Sinclair: "After all that has happened you really think I will join you?".
- //nobody ever wants to be on my side, even in TEAM fortress I am on my own- Unknown Voice\\
- //(cant think of anything)\\
- Unknown Voice: "You will Join me or you will perish! STOP HIM!!!".

Scene 2

The Player enters the Server Room only to get a surprise.

- Unknown Voice: "You are better than I gave you credit for Professor".
- Sinclair: "You have nowhere to go, just give yourself up!".
- ATHENA: " Sinclair doesn't he seem familiar?".
- Unknown Voice: "Trust a stupid AI to figure it out before our esteemed Professor Sinclair Rogey boy would be happy how clever she is".
- Sinclair: "GRR Shut it! You Murderer! YOU KILLED MY BEST FRIEND!".
- ATHENA: "Sinclair lis..."

- UV: "Hahah best friend looks like you really did not know much about him, now would you kindly die Professor Mikkel Sinclair".
- <<Unknown Voice turns around to >>
- Sinclair: "it can't be, how no...no NOOOOO".
- ATHENA: "Sinclair snap out of it we need to stop him".
- Rogers: "Stop me hahaha you cant stop me!".
- Sinclair: "Why, why did you do this why?".

<<without warning Rogers start attacking>>

<<epic battle sequence>>

Scene 3

After an epic battle Sinclair finally wins and before Rogers dies Sinclair talks

- Sinclair: "Why did you do this Adrian, we were friends we were going to make a difference".
- Rogers: " Haha can you not see what we have created Mikkel?".
- Sinclair: " Yes away to better humanity".
- Rogers: " You naïve fool haha * cough* what you have created are weapons, just look at what you have done, killed many men. Our creations that's all theu will do, destroy".
- Sinclair: " Yes they have destructive properties, but we have created many great ways to use it to better humanity, I can easily get rid of the weaponised parts".
- Rogers: " Haha *cough* after all this do you really think you can let go off all the power you have, and do you really think this humanity you believe in will take this new technology as a gift and use it for the better haha *cough* haha I will leave the decision up to you my old friend".

<< Rogers passes on>>

- ATHENA: "Sinclair I am sorry for what you had to do".
- Sinclair: "its fine ATHENA it happened because the thought of power was over wellming for him".
- ATHENA: "What now".

<<Player has three choices>>

Good Choice

- Sinclair: " I am going to create a better Future for all of humanity, I will get rid of all weponised Augments and show the world that they can save many, but this will also mean that you will have to be deactivated ATHENA".
- ATHENA: "Professor I... Sinclair is was an honor known sir".

Bad Choice

- Sinclair: " Adrian was right these 'weapons' are too powerful to let Civilians get their hands on them".
- ATHENA: "Sinclair what do you mean?">
- Sinclair: " The power of Augments deserve to be used properly, their full Potential will never be reconnised if I don't use them to better our country, and I can by do that my allowing them to be used in the hands of the military!".
- ATHENA: "Sinclair you can't mean that".
- Sinclair: " ATHENA you were created to help me control the Augs and to listen to my every COMMAND!".
- ATHENA: "Yes Sir I understand...".

Neutral Choice

- Sinclair: " What have I done...?, I killed my Best Friend, I have killed so many people using my creations that were to be used to help better mankind".

- ATHENA: " And that's what you have done! Sinclair they would have used these Augmentations to kill millions of people, you stopped the bad guys so you can save mankind and help make it better".
- Sinclair: " What are you talking about, my creations were never meant to be used this way! Adrian, o god Adrian he was right! These weapons are far too dangerous, what have I done! Science is used to help mankind and save life's, not to destroy it!!! O god what have I done".
- ATHENA: " PROFESSOR CALM DOWN! It's not your fault you've done what you had to do!".
- Sinclair: " No your wrong I cant let this kind of future happen it will destroy humanity instead of save it! I cant let my creations see the light of day I cant! Goodbye ATHENA..."

EPILOGUE

Good Choice

- After all the Chaos that Sinclair endured and caused, just to keep his creations from the military, made him realize that the Weaponised Augmentations were far too dangerous to be mass produced, even with the help of ATHENA managing everything. After a tough decision Sinclair decided to create the Augments to tailor around helping Humanity without weapons. This however did not prevent some casualties as many men who were just under orders, had to die just for the Great Professor to realize this, he also lost his dear companion and Friend ATHENA in the process, to help prevent the Augments to be used in a military manner.

Bad Choice

- After all the Chaos that Sinclair endured and caused, just to keep his creations from the military and the death of a Friend who made the wrong choices, the great Professor had come to a realization that his creations were created for the purpose he himself had used them, to cause death and destruction, he thought about it and controlling the military by letting them use his augments meant he could save many from the horrors of war and could help save the Soldiers that fought in them. Many thought his ideas were crazy and that he ruined his ideals of saving humanity. But the horrors of what occurred and with the help of his helper Athena, Sinclair changed and a more colder and darker Sinclair appeared, one who just wanted to save lives by causing death and destruction.

Neutral Choice

- After all the Chaos that Sinclair endured and caused, just to keep his creations from the military, It became too much for the poor Professor, he went into a state of shock over the carnage he had caused. He didn't understand how he could continue on known that his lifes work in making humanity better would cause death and destruction. Even ATHENA the AI he had come to love as a friend could not help him out of his shell. Without out a plan to continue he had decided on the only course of action he thought would help. He blew up the Facility and in the process killed himself and the lovely ATHENA.

If you were Sinclair would you have made different choices, would you have changed the outcome of what has happened?

2 Appendix 2 - Genre

Genre

Computing Project

Project Plan

Interactive Story / Hybrid RPG (To The Moon, Final Fantasy)

Platformer (Thomas Was Alone)

Puzzle (Bejeweled)

Turn-based Strategy (Civilization 5)

Video Board Game (Monopoly)

Text-based game (Zork)

Shooter (Space Invaders)

Racing ()

On-Rails game ()

Tycoon (Rollercoaster Tycoon)

Tycoon - Money, buy/sell, market, customer, product (possibly custom), characters, competitors, possibly a goal/goals, locations

RPG - Story, loot, enemies, characters, levelling, stats, world map, towns, character design, level design, bosses,

Votes

Kieran – Hybrid RPG, Tycoon

Grant – Hybrid RPG, Turn-Based or Tycoon

Alastair – Top-down shooter, Tycoon game

Ideas

Game Development tycoon

Mafia/Crime Tycoon

Video/Comic Book Tycoon

Cinema Tycoon

Business Tycoon

Farming Tycoon

Dungeon-Based RPG

Science Fiction RPG

3 Appendix 3 - Characters

Characters

Wednesday, February 5, 2014

1:34 PM

The Computer AI Assistant	ATHENA - Automated Tactical Heads-Up and Environment Navigation Aid
The protagonist	Mikkel Sinclair (Professor Sinclair)
The research assistant	Adrian Rogers (Dr. Rogers)
Military commander	Commander Ethan Kemp
Science facility	Augmentation Research Centre (ARC)

4 Appendix 4 - Levelling

There are two main skill trees, Military and Humanity. The choices affect the overall ending depend on which traits they choose, there will be a total of 10 levels each with 2 different choices, one favouring humanity and the other Military. The ending of the game will reflect the skills chosen by the player. If the player chooses an equal number of both military and humanity then the game ending will be neutral.

Humanity Skills

- 1: Nano Basic Heal – Heals 25% of health
- 2: Nano Heal – Heals 50% of health
- 3: Nano Advanced Heal – Heals 75% of health
- 4: Nano Basic Protect – Damage is reduced by 25%
- 5: Nano Protect – Damage is reduced by 50%
- 6: Nano Advanced Protect - Damage is reduced by 75%
- 7: Basic Health node – HP is increased by 25%
- 8: Health node – HP is increased by 50%
- 9: Basic Nano capacity – Increase Nano storage by 25%
- 10: Nano Shield – No damage for turn.

Weaponisation

- 1: Basic Damage Increase – Damage is Increased by 25%
- 2: Damage Increase – Damage is increased by 50%
- 3: Decimation – Damage is increased by 75%
- 4: Basic Thick Skin – Damage is reduced by 25%
- 5: Thick Skin – Damage is reduced by 50%
- 6: Ballistic Skin – Damage is reduced by 75%
- 7: Nano Drain – Health is restored by 25% at the end of each battle
- 8: Advanced Nano Drain - Health is restored by 50% at the end of each battle
- 9: Nano Pulse Cannon – A laser with the Damage of 100
- 10: Kamehameha – Instakill

5 Appendix 5 - RPG Plan

RPG Ideas

Wednesday, February 5, 2014

11:41 AM

Viewpoints

Side On
From Above
Perspective

World Design

One big dungeon

Story

Setting - Near Future (Deus Ex / Anarchy Online)

World

Different world.

Plot

Nano-Aug human - scientist that invented augmentations, wants to give people the augmentations to make the world better but the military try to steal it for weapons. The government-funded facility/project is attacked by the military, the scientist has to fight through the army and learn how to use his augmentations to fight. He has to fight though the facility. He has to choose whether to go the route of "weaponization" vs "humanity" and the game ending will reflect choices made. The game would have three endings based on the augmentation choices. "Good" or "for the benefit of humanity" ending where the augmentations benefit everyone as a whole, "bad" or the "military" ending, where the augmentations are designed for warfare, or "neutral", where the augmentation project is destroyed. The final boss is a research assistant who wants to keep the research for himself. (A God Am I applies).

6 Appendix 6 - Items/Experience Plan

Items – There will be very few items in the game mostly to refill health/nano. There will be a couple of Main Items relating to the story

Key Items

Server Room Key

ATHENA

Items

Potion – Heals 25% of HP

Mega-potion – Heals 50% of HP

Max-potion – Heals 100% HP

Nano-Potion – Refills 25% Nano

Mega-Nano – Refills 50% Nano

Max-Nano – Refills 100% Nano

Experience

Soldier – 25EXP

Elite – 50EXP

Commander – 100EXP

BOSS 1 – 500EXP

Rodgers – 1000EXP

Level EXP

Level 1 – 150

Level 2 – 300

Level 3 – 500

Level 4 – 750

Level 5 – 1000

Level 6 - 1250

Level 7 - 1500

Level 8 - 1750

Level 9 - 2000

Level 10 - 2500

7 Appendix 7 - Sprites/Graphics

7.1 Characters

7.1.1 Professor Sinclair



7.1.2 Professor's Assistant



7.1.3 Military Commander



7.1.4 Soldier



7.1.5 Elite Soldier



7.1.6 Security Guard

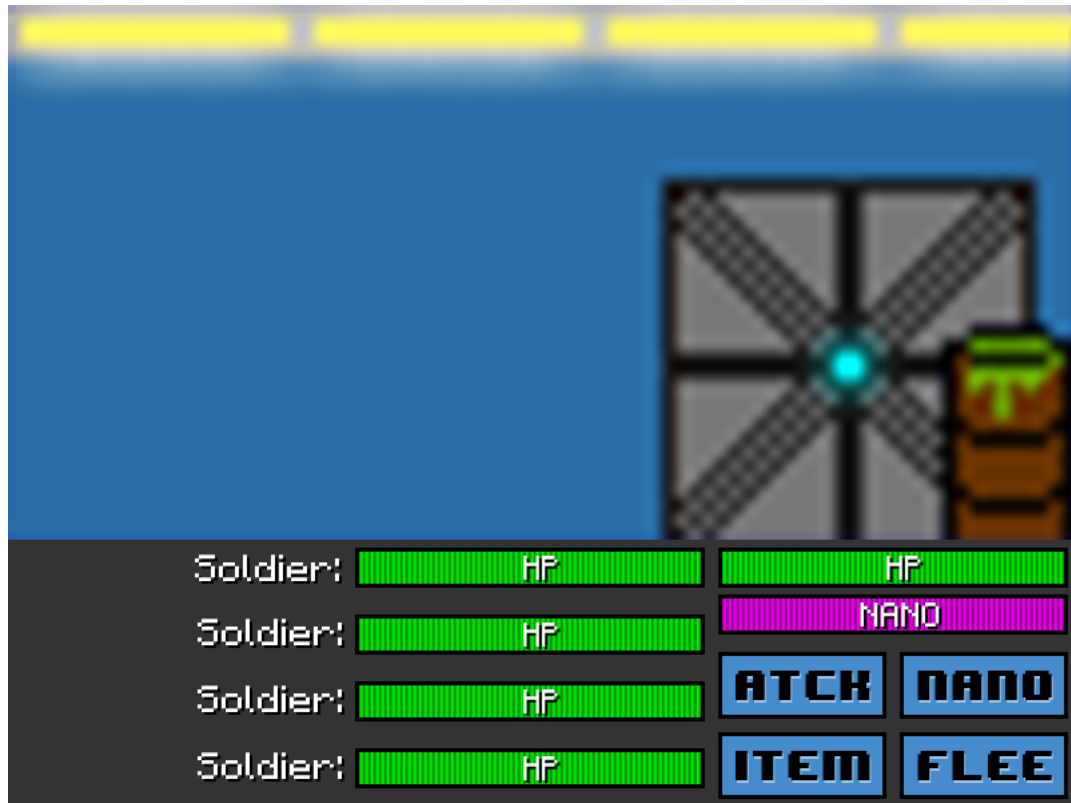


7.2 Other

7.2.1 Battle background



7.2.2 Battle UI



8 Appendix 8 - Code

Athena

Assembly Info

```
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

// General Information about an assembly is controlled through the following
// set of attributes. Change these attribute values to modify the information
// associated with an assembly.
[assembly: AssemblyTitle("Athena")]
[assembly: AssemblyProduct("Athena")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyCopyright("Copyright © 2014")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]

// Setting ComVisible to false makes the types in this assembly not visible
// to COM components. If you need to access a type in this assembly from
// COM, set the ComVisible attribute to true on that type.
[assembly: ComVisible(false)]

// The following GUID is for the ID of the typelib if this project is exposed to COM
[assembly: Guid("ef85f802-4875-4a87-92d1-e00ec9452bcc")]

// Version information for an assembly consists of the following four values:
//
//      Major Version
//      Minor Version
//      Build Number
//      Revision
//
// You can specify all the values or you can default the Build and Revision Numbers
// by using the '*' as shown below:
// [assembly: AssemblyVersion("1.0.*")]
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]
```

Levels

```
?xml version="1.0" encoding="utf-8" ?>
<level tileset="pokemon" tileSize="25" rowlength="3">
  <tile>
    <sprite>paving.cells</sprite>
  </tile>
  <tile>
    <sprite>paving.cells</sprite>
  </tile>
  <tile>
    <sprite>black</sprite>
  </tile>
  <tile>
    <sprite>black</sprite>
  </tile>
</level>
```

UI Font

```
<?xml version="1.0" encoding="utf-8"?>
<!--
This file contains an xml description of a font, and will be read by the XNA
Framework Content Pipeline. Follow the comments to customize the appearance
of the font in your game, and to change the characters which are available to draw
with.
-->
<XnaContent xmlns:Graphics="Microsoft.Xna.Framework.Content.Pipeline.Graphics">
  <Asset Type="Graphics:FontDescription">

    <!--
    Modify this string to change the font that will be imported.
    -->
    <FontName>Minecraftia</FontName>

    <!--
    Size is a float value, measured in points. Modify this value to change
    the size of the font.
    -->
    <Size>18</Size>

    <!--
    Spacing is a float value, measured in pixels. Modify this value to change
    the amount of spacing in between characters.
    -->
    <Spacing>3</Spacing>

    <!--
    UseKerning controls the layout of the font. If this value is true, kerning
information
    will be used when placing characters.
    -->
    <UseKerning>>false</UseKerning>

    <!--
    Style controls the style of the font. Valid entries are "Regular", "Bold",
    "Italic",
    and "Bold, Italic", and are case sensitive.
    -->
```

```

<Style>Regular</Style>

<!--
If you uncomment this line, the default character will be substituted if you draw
or measure text that contains characters which were not included in the font.
-->
<!-- <DefaultCharacter>*</DefaultCharacter> -->

<!--
CharacterRegions control what letters are available in the font. Every
character from Start to End will be built and made available for drawing. The
default range is from 32, (ASCII space), to 126, ('~'), covering the basic Latin
character set. The characters are ordered according to the Unicode standard.
See the documentation for more information.
-->
<CharacterRegions>
  <CharacterRegion>
    <Start>&#32;</Start>
    <End>&#126;</End>
  </CharacterRegion>
</CharacterRegions>
</Asset>
</XnaContent>

```

Tileset

```

<?xml version="1.0" encoding="utf-8" ?>
<tileset tilesize="25">
  <sprite>
    <name>black</name>
    <x>0</x>
    <y>0</y>
    <collides>false</collides>
  </sprite>
  <sprite>
    <name>tile.diagonal</name>
    <x>1</x>
    <y>0</y>
    <collides>false</collides>
  </sprite>
  <sprite>
    <name>carpet.horizontal</name>
    <x>2</x>
    <y>0</y>
    <collides>false</collides>
  </sprite>
  <sprite>
    <name>paving.cells</name>
    <x>2</x>
    <y>0</y>
    <collides>false</collides>
  </sprite>
</tileset>

```


Debug

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Athena
{
    public static class Debug
    {
        public static Modes Mode;
        public enum Modes { GAME, UI };
    }
}
```

Game1

```
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;

using AthenaEngine.Framework;
using AthenaEngine.Framework.Gameplay;
using AthenaEngine.Framework.Interfaces;
using AthenaEngine.Framework.Primatives;
using AthenaEngine.Framework.Systems;
using AthenaEngine.Framework.UI;

namespace Athena
{
    /// <summary>
    /// This is the main type for your game
    /// </summary>
    public class Game1 : Microsoft.Xna.Framework.Game
    {
        // TODO: Remove warning suppression once it's used.
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Performance",
"CA1823:AvoidUnusedPrivateFields")]
        GraphicsDeviceManager graphics;
        SpriteBatch spriteBatch;
        ResourceManager<Texture2D> textureManager;
        ResourceManager<SpriteFont> fontManager;
        Level test;
        Camera2D Camera;
        Character Player;
        bool FirstRun = true;
        KeyboardState OldState;

        /// <summary>
        /// The actual game class constructor.
    }
}
```

```

    /// </summary>
    public Game1()
    {
        graphics = new GraphicsDeviceManager(this);
        Content.RootDirectory = "Content";
    }

    /// <summary>
    /// Allows the game to perform any initialization it needs to before starting
to run.
    /// This is where it can query for any required services and load any non-
graphic
    /// related content. Calling base.Initialize will enumerate through any
components
    /// and initialize them as well.
    /// </summary>
    protected override void Initialize()
    {
        // TODO: Add your initialization logic here

        base.Initialize();
    }
    public SpriteFont font;

    /// <summary>
    /// LoadContent will be called once per game and is the place to load
    /// all of your content.
    /// </summary>
    protected override void LoadContent()
    {
        // Create a new SpriteBatch, which can be used to draw textures.
        spriteBatch = new SpriteBatch(GraphicsDevice);
        textureManager = new ResourceManager<Texture2D>(this);
        fontManager = new ResourceManager<SpriteFont>(this);

        this.Camera = new Camera2D(this);

        textureManager.Add("blank", this.Content.Load<Texture2D>("blank"));
        textureManager.Add("pokemon", this.Content.Load<Texture2D>("pokemon-
tiles"));
        textureManager.Add("Boards", this.Content.Load<Texture2D>("Boards"));
        textureManager.Add("guy", this.Content.Load<Texture2D>("guy"));
        fontManager.Add("SpriteFont1",
this.Content.Load<SpriteFont>("SpriteFonts/UIFont"));
        test = new Level("level1", spriteBatch, textureManager);
        this.Player = new Character(new Vector2(75, 75), new Vector2(25, 25),
spriteBatch, textureManager.Get("guy"), test);
        // Player.SpriteColor = Color.Red;
        Camera.Focus = Player;
        Camera.Initialize();
    }

    /// <summary>
    /// UnloadContent will be called once per game and is the place to unload
    /// all content.
    /// </summary>
    protected override void UnloadContent()
    {
        // TODO: Unload any non ContentManager content here
    }

    /// <summary>

```

```

/// Allows the game to run logic such as updating the world,
/// checking for collisions, gathering input, and playing audio.
/// </summary>
/// <param name="gameTime">Provides a snapshot of timing values.</param>
protected override void Update(GameTime gameTime)
{
    KeyboardState State = Keyboard.GetState();

    if (FirstRun)
    {
        if (State.IsKeyDown(Keys.W))
        {
            Player.Move(Directions.UP);
        }
        if (State.IsKeyDown(Keys.A))
        {
            Player.Move(Directions.LEFT);
        }
        if (State.IsKeyDown(Keys.S))
        {
            Player.Move(Directions.DOWN);
        }
        if (State.IsKeyDown(Keys.D))
        {
            Player.Move(Directions.RIGHT);
        }

        OldState = State;
        FirstRun = false;
    }
    else if (OldState != State)
    {
        if (State.IsKeyDown(Keys.W))
        {
            Player.Move(Directions.UP);
        }
        if (State.IsKeyDown(Keys.A))
        {
            Player.Move(Directions.LEFT);
        }
        if (State.IsKeyDown(Keys.S))
        {
            Player.Move(Directions.DOWN);
        }
        if (State.IsKeyDown(Keys.D))
        {
            Player.Move(Directions.RIGHT);
        }

        OldState = State;
    }
    else
    {
        // Do nothing.
    }

    Camera.Update(gameTime);
    base.Update(gameTime);
}

/// <summary>
/// This is called when the game should draw itself.

```

```

    /// </summary>
    /// <param name="gameTime">Provides a snapshot of timing values.</param>
    protected override void Draw(GameTime gameTime)
    {
        Debug.Mode = Debug.Modes.GAME;

        GraphicsDevice.Clear(Microsoft.Xna.Framework.Color.CornflowerBlue);

        if (Debug.Mode == Debug.Modes.GAME)
        {
            Camera.Focus = Player;

            spriteBatch.Begin(SpriteSortMode.Deferred, BlendState.AlphaBlend,
null, null, null, null, Camera.Transform);

            test.Draw();
            Player.Draw();
            spriteBatch.End();
        }

        else if (Debug.Mode == Debug.Modes.UI)
        {
            //          UIButton test = new UIButton(new Vector2(0, 0), new Vector2(50, 50),
spriteBatch, textureManager.Get("blank"));
            UI Test = new UI(spriteBatch, textureManager, fontManager);

            Test.AddButton(new Vector2(5, 5), "HI");
            Test.AddButton(new Vector2(50, 50), "ASK!");
            spriteBatch.Begin();
            // Test.Draw();
            // int x = 6;
            // int y = 105;

            // spriteBatch.Draw(textureManager.Get("pokemon"), new Rectangle(0, 0,
25, 25), new Rectangle((x - 1) * 25, (y - 1) * 25, 25, 25), Color.White);
            Player.Draw();
            test.Draw();

            spriteBatch.End();

        }

        base.Draw(gameTime);
    }
}

```

Program

```
#region Using Statements
using System;
using System.Collections.Generic;
using System.Linq;
#endregion

namespace Athena
{
    #if WINDOWS || LINUX
        /// <summary>
        /// The main class.
        /// </summary>
        public static class Program
        {
            /// <summary>
            /// The main entry point for the application.
            /// </summary>
            [STAThread]
            static void Main()
            {
                using (var game = new Game1())
                    game.Run();
            }
        }
    #endif
}
```

Athena Engine

RPG

Inventory

```
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;

namespace AthenaEngine.Framework.Gameplay.RPG
{
    /// <summary>
    /// Inventory.
    /// </summary>
    class Inventory
    {
        private List<ItemInstance> ItemList = new List<ItemInstance>();
        private int ItemCount;
```

```

        /// <summary>
        /// Add the specified item and quantity.
        /// </summary>
        /// <param name='item'>
        /// Item.
        /// </param>
        /// <param name='quantity'>
        /// Quantity.
        /// </param>
        public void Add(Item item, int quantity)
        {
            this.ItemList.Add(new ItemInstance(item, quantity));
        }
    }
}

```

Item

```

using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;

namespace AthenaEngine.Framework.Gameplay.RPG
{
    /// <summary>
    /// A gameplay item can be held by a character.
    /// </summary>
    public class Item
    {
        /// <summary>
        /// The name of the item
        /// </summary>
        public string Name {
            get { return this.Name; }
            private set { this.Name = value; }
        }

        /// <summary>
        /// Constructor for the item class
        /// </summary>
        /// <param name="name">Name of the item</param>
        public Item(string name)
        {
            this.Name = name;
        }
    }
}

```

Item Instance

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace AthenaEngine.Framework.Gameplay.RPG
{
    /// <summary>
    /// An item instance is a particular instance of an item.
    /// </summary>
    public class ItemInstance : Item
    {
        private int Quantity;
        /// <summary>
        /// Constructor for the ItemInstance class.
        /// </summary>
        /// <param name="item">The actual item of which this is an instance</param>
        /// <param name="quantity">How many of that item are in this particular
instance?</param>
        public ItemInstance(Item item, int quantity) : base(item.Name)
        {
            this.Quantity = quantity;
        }
    }
}
```

Character

```
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;

using AthenaEngine.Framework.Primitives;
using AthenaEngine.Framework.Interfaces;

namespace AthenaEngine.Framework.Gameplay
{
    /// <summary>
    /// A character class holds important detail about each character such as their
items, level, experience, skills, etc.
    /// </summary>
    public class Character : DrawableEntity, IFocusable
    {
        private int Level;
        private int Experience;

        /// <summary>
        /// Constructor for the Character class
        /// </summary>
        /// <param name="position">The coordinates of the character</param>
        /// <param name="size">The size of the character</param>
    }
}
```

```

        /// <param name="spriteBatch">Which spriteBatch will draw the
character</param>
        /// <param name="texture">What is the texture for the character</param>
        public Character(Vector2 position, Vector2 size, SpriteBatch spriteBatch,
Texture2D texture, Level level) : base(position, size, new Rectangle(0, 0, 25, 25),
spriteBatch, texture, level)
        {

        }

    }
}

```

Level

```

using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;

using AthenaEngine.Framework.Systems;

namespace AthenaEngine.Framework.Gameplay
{
    /// <summary>
    /// A Level object holds all details required to handle level drawing.
    /// </summary>
    public class Level
    {
        public List<Tile> TileList;

        /// <summary>
        /// Object constructor for the Level class.
        /// </summary>
        /// <param name="levelName">The name of the level to load.</param>
        /// <param name="spriteBatch">The spriteBatch to draw the level with.</param>
        /// <param name="resourceManager">The resourceManager handling the games'
textures.</param>
        public Level(string levelName, SpriteBatch spriteBatch,
ResourceManager<Texture2D> resourceManager)
        {
            this.TileList = LevelLoaderXml.Load(levelName, this);

            foreach (Tile tile in TileList)
            {
                tile.MakeDrawable(spriteBatch, resourceManager);
            }

            /// <summary>
            /// Draw the level.
            /// </summary>
            public void Draw()
            {

```



```

        foreach (Tile tile in TileList)
        {
            tile.Draw();
        }
    }
}

```

Tile

```

using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;
using System.Reflection;

using AthenaEngine.Framework.Primitives;
using AthenaEngine.Framework.Interfaces;
using AthenaEngine.Framework.Systems;

namespace AthenaEngine.Framework.Gameplay
{
    /// <summary>
    /// A tile is used to draw levels.
    /// </summary>
    public class Tile : CollidableEntity
    {
        private bool Drawable;
        public bool Collides = false;
        private Texture2D Texture;
        public DrawableEntity Sprite;
        private string TileSet;
        private Rectangle SourceRectangle;
        private MethodInfo Trigger;
        public bool HasTrigger = false;

        /// <summary>
        /// Class constructor for a new Tile object.
        /// </summary>
        /// <param name="x">The X coordinate of the new tile.</param>
        /// <param name="y">The Y coordinate of the new tile.</param>
        public Tile(int x, int y, string TileSet, int xOffset, int yOffset, Level
level) : base(new Vector2(x, y), new Vector2(25, 25), level)
        {
            this.TileSet = TileSet;
            this.SourceRectangle = new Rectangle(xOffset, yOffset, 25, 25);
        }

        /// <summary>
        /// Make a tile drawable by giving it a texture and a spriteBatch.
        /// </summary>
        /// <param name="spriteBatch">The SpriteBatch to draw the tile with.</param>
        /// <param name="texture">The texture to draw the tile with.</param>
        public void MakeDrawable(SpriteBatch spriteBatch, ResourceManager<Texture2D>
resoureManager)

```

```

    {
        this.Drawable = true;

        this.Texture = resoureManager.Get(TileSet);

        this.Sprite = new DrawableEntity(Position, Size, SourceRectangle,
spriteBatch, this.Texture, this.Level);
    }

    /// <summary>
    /// Draw the tile.
    /// </summary>
    public void Draw()
    {
        if (this.Drawable)
        {
            this.Sprite.Draw();
        }
    }

    /// <summary>
    /// Fires the trigger.
    /// </summary>
    /// <param name='args'>
    /// Arguments.
    /// </param>
    public void FireTrigger (object[] args)
    {
        if (this.HasTrigger)
        {
            Trigger.Invoke(null, args);
        }
    }

    /// <summary>
    /// Adds the trigger.
    /// </summary>
    /// <param name='triggerName'>
    /// Trigger name.
    /// </param>
    public void AddTrigger(string triggerName)
    {
        Type type = typeof(Triggers);
        MethodInfo method = type.GetMethod(triggerName);
        this.Trigger = method;
        this.HasTrigger = true;
    }
}
}

```

I Collidable

```

using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;

```

```

using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;

using AthenaEngine.Framework.Primitives;

namespace AthenaEngine.Framework.Interfaces
{
    /// <summary>
    /// I collidable.
    /// </summary>
    interface ICollidable<T>
    {
        bool CollidesWith(T type);
    }
}

```

I Drawable

```

using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;

namespace AthenaEngine.Framework.Interfaces
{
    /// <summary>
    /// I drawable.
    /// </summary>
    interface IDrawable
    {
        /// <summary>
        /// Draw this instance.
        /// </summary>
        void Draw();
    }
}

```

I Focusable

```

using System;
using System.Collections.Generic;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;

namespace AthenaEngine.Framework.Interfaces
{

```

```

    /// <summary>
    /// I focusable.
    /// </summary>
    public interface IFocusable
    {
        /// <summary>
        /// Gets the position.
        /// </summary>
        /// <value>
        /// The position.
        /// </value>
        Vector2 Position { get; }
    }
}

```

I Moveable

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace AthenaEngine.Framework.Interfaces
{
    interface IMoveable
    {
        bool Move(string direction);
        bool CanMove(string direction);
    }
}

```

Bounding Box 2D

```

using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;

namespace AthenaEngine.Framework.Primitives
{
    /// <summary>
    /// BoundingBox2D is used for bounding boxes on 2D objects.
    /// </summary>
    public class BoundingBox2D : Entity, IEquatable<BoundingBox2D>
    {
        protected Rectangle Bounds
        {
            get
            {
                return this.Rectangle;
            }
        }
    }
}

```

```

        set
        {
            this.Rectangle = value;
        }
    }
    /// <summary>
    /// Constructor for BoundingBox2D
    /// </summary>
    /// <param name="min">x/y coordinates for the bounding box</param>
    /// <param name="max">width/height for the bounding box</param>
    public BoundingBox2D(Vector2 Position, Vector2 Size)
    {
        base.Position = Position;
        base.Size = Size;
    }

    /// <summary>
    /// Check if the BoundingBox2D is equal to another BoundingBox2D
    /// </summary>
    /// <param name="otherBounds">The other BoundingBox2D to check
against.</param>
    /// <returns></returns>
    public bool Equals (BoundingBox2D otherBounds)
    {
        if (this.Bounds.Equals(otherBounds.Bounds))
        {
            return true;
        }
        else
        {
            return false;
        }
    }

    /// <summary>
    /// Check to see if this BoundingBox2D collides with an other BoundingBox.
    /// </summary>
    /// <param name="otherBounds">The other BoundingBox2D to compare with.</param>
    /// <returns></returns>
    public bool CollidesWith(BoundingBox2D other)
    {
        bool Colliding = false;

        if (this.Equals(other))
        {
            Colliding = true;
        }
        if ((this.Bounds.Bottom < other.Bounds.Bottom) && (this.Bounds.Bottom >
other.Bounds.Top))
        {
            Colliding = true;
        }
        if ((this.Bounds.Top > other.Bounds.Top) && (this.Bounds.Top <
other.Bounds.Bottom))
        {
            Colliding = true;
        }
        if ((this.Bounds.Left > other.Bounds.Left) && (this.Bounds.Left <
other.Bounds.Right))
        {
            Colliding = true;
        }
    }

```

```

        }
        if ((this.Bounds.Right < other.Bounds.Right) && (this.Bounds.Right >
other.Bounds.Left))
        {
            Colliding = true;
        }

        return Colliding;
    }
}

```

Collidable Entity

```

using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;

using AthenaEngine.Framework.Interfaces;
using AthenaEngine.Framework.Gameplay;

namespace AthenaEngine.Framework.Primitives
{
    /// <summary>
    /// The Entity class is used to store objects that have positions.
    /// </summary>
    public abstract class CollidableEntity : Entity
    {
        protected BoundingBox2D Bounds
        {
            get
            {
                return new BoundingBox2D(Position, Size);
            }
            set
            {
                if (this.Size.X < 0 || this.Size.Y < 0)
                {
                    throw new ArithmeticException("Object size is invalid, less than
0.");
                }
                this.Position = new Vector2(value.X, value.Y);
                this.Size = new Vector2(value.Width, value.Height);
            }
        }

        public Level Level;

        /// <summary>
        /// Constructor for the CollidableEntity class.
        /// </summary>
        /// <param name="position">x/y coordinates of the entity.</param>
        /// <param name="size">width/height of the entity.</param>
        public CollidableEntity(Vector2 position, Vector2 size, Level level)
    }
}

```

```

{
    this.Position = position;
    this.Size = size;
    this.Bounds = new BoundingBox2D(position, size);
    this.Level = level;
}

/// <summary>
/// Check if the CollidableEntity collides with another CollidableEntity.
/// </summary>
/// <param name="entity">The entity to check against</param>
/// <returns>Returns true if it does collide, otherwise false.</returns>
public bool CollidesWith (CollidableEntity entity)
{
    if (this.Bounds.CollidesWith(entity.Bounds))
    {
        return true;
    }
    else
    {
        return false;
    }
}

/// <summary>
/// Determines whether this instance can move the specified direction.
/// </summary>
/// <returns>
/// <c>true</c> if this instance can move the specified direction;
otherwise, <c>false</c>.
/// </returns>
/// <param name='direction'>
/// If set to <c>true</c> direction.
/// </param>
public bool CanMove(int direction)
{
    BoundingBox2D NewPosition = new BoundingBox2D(new Vector2(this.Bounds.X,
this.Bounds.Y), new Vector2(this.Bounds.Width, this.Bounds.Height));

    switch (direction)
    {
        case Directions.UP:
            NewPosition.Y -= 25;
            break;
        case Directions.DOWN:
            NewPosition.Y += 25;
            break;
        case Directions.LEFT:
            NewPosition.X -= 25;
            break;
        case Directions.RIGHT:
            NewPosition.X += 25;
            break;
    }
    foreach (Tile t in Level.TileList)
    {
        if (t.Collides)
        {
            if (NewPosition.CollidesWith(t.Bounds))
            {
                return false;
            }
        }
    }
}

```

```

        else
        {
            // There wasn't.
        }
    }
}

return true;
}

/// <summary>
/// Move the specified direction.
/// </summary>
/// <param name='direction'>
/// If set to <c>true</c> direction.
/// </param>
public bool Move(int direction)
{
    if (CanMove(direction))
    {
        base.Move(direction);
        foreach (Tile t in Level.TileList)
        {
            if ((t.X == this.X) && (t.Y == this.Y))
            {
                t.FireTrigger(null);
            }
        }
        return true;
    }
    else
    {
        System.Console.WriteLine("Collision detected.");
        return false;
    }
}
}
}
}

```

Directions

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace AthenaEngine.Framework.Primitives
{
    public abstract class Directions
    {
        public const int LEFT = 0;
        public const int UP = 1;
        public const int RIGHT = 2;
        public const int DOWN = 3;
    }
}

```


Drawable Entity

```
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;

using AthenaEngine.Framework.Gameplay;

namespace AthenaEngine.Framework.Primitives
{
    /// <summary>
    /// This is an entity which can be drawn.
    /// </summary>
    public class DrawableEntity : CollidableEntity, Interfaces.IDrawable,
    Interfaces.IFocusable
    {
        public Color SpriteColor;

        protected SpriteBatch SpriteController;
        protected Texture2D SpriteSheet;
        protected Rectangle SpriteSource;
        public Level level;

        /// <summary>
        /// This is the constructor for the DrawableEntity class.
        /// </summary>
        /// <param name="position">Where the DrawableEntity will start</param>
        /// <param name="size">The size in pixels of the DrawableEntity</param>
        /// <param name="spriteBatch">The SpriteBatch responsible for drawing the
entity</param>
        /// <param name="texture">The texture used to draw the entity</param>
        public DrawableEntity(Vector2 position, Vector2 size, Rectangle SpriteSource,
SpriteBatch spriteBatch, Texture2D tileset, Level level) : base(position, size, level)
        {
            this.SpriteSource = SpriteSource;
            this.SpriteColor = Color.White;
            this.SpriteSheet = tileset;
            this.Position = position;
            this.Size = size;
            this.SpriteController = spriteBatch;
        }

        /// <summary>
        /// This draws the DrawableEntity.
        /// </summary>
        public void Draw()
        {
            this.SpriteController.Draw(this.SpriteSheet, this.Rectangle,
this.SpriteSource, this.SpriteColor);
        }
    }
}
```

```

        public Vector2 Position
        {
            get
            {
                return base.Position;
            }
            set
            {
                base.Position = value;
            }
        }
    }
}

```

Entity

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;

using AthenaEngine.Framework.Primitives;

namespace AthenaEngine.Framework.Primitives
{
    /// <summary>
    /// The Entity class is the superclass for anything.
    /// </summary>
    public abstract class Entity
    {
        protected Vector2 Position;
        protected Vector2 Size;

        /// <summary>
        /// Gets or sets the rectangle.
        /// </summary>
        /// <value>
        /// The rectangle.
        /// </value>
        protected Rectangle Rectangle
        {
            get
            {
                return new Rectangle(X, Y, Width, Height);
            }
            set
            {
                this.Position = new Vector2(value.X, value.Y);
                this.Size = new Vector2(value.Width, value.Height);
            }
        }
    }
}

```

```

    }

    public int X
    {
        get
        {
            return (int) Position.X;
        }
        set
        {
            Position.X = value;
        }
    }
    public int Y
    {
        get
        {
            return (int)Position.Y;
        }
        set
        {
            Position.Y = value;
        }
    }
    public int Width
    {
        get
        {
            return (int)Size.X;
        }
        set
        {
            Size.X = value;
        }
    }
    public int Height
    {
        get
        {
            return (int)Size.Y;
        }
        set
        {
            Size.Y = value;
        }
    }
}

protected void Move(int direction)
{
    switch (direction)
    {
        case Directions.UP:
            this.Y -= 25;
            break;
        case Directions.DOWN:
            this.Y += 25;
            break;
        case Directions.LEFT:
            this.X -= 25;
            break;
        case Directions.RIGHT:
            this.X += 25;
    }
}

```

```

                break;
            default:
                throw new InvalidOperationException("Invalid direction to move
in");
        }
    }
}

```

Level Loader XML

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using System.Xml.Linq;
using System.Reflection;

using AthenaEngine.Framework.Gameplay;

namespace AthenaEngine.Framework.Systems
{
    /// <summary>
    /// the LevelLoader is used to load levels.
    /// </summary>
    public static class LevelLoaderXml
    {
        public class Sprite
        {
            public int X;
            public int Y;
            public string Name;
            public bool Collides;

            public Sprite(int x, int y, string name, bool collides)
            {
                this.X = x;
                this.Y = y;
                this.Name = name;
                this.Collides = collides;
            }
        }

        /// <summary>
        /// Load the specified levelName and level.
        /// </summary>
        /// <param name='levelName'>
        /// Level name.
        /// </param>
        /// <param name='level'>
        /// Level.
        /// </param>
        public static List<Tile> Load (string levelName, Level level)
        {
            List<Tile> LevelList = new List<Tile>();
            XDocument Document = XDocument.Load("Content/Levels/" + levelName +
"/tiles.xml");
            XElement Level = Document.Element("level");
            string Tilesset = Level.Attribute("tilesset").Value;

```

```

        int RowLength = Int32.Parse(Level.Attribute("rowlength").Value);

        Dictionary<string, Sprite> SpriteList = LoadSpriteSet(Tileset);

        System.Console.WriteLine(Tileset);
        int TileSize = Int32.Parse(Level.Attribute("tilesize").Value);

        IEnumerable<XElement> Tiles = Level.Elements("tile");
        int Y = 0;
        int X = 0;
        foreach (XElement Tile in Tiles)
        {
            string SpriteName = Tile.Element("sprite").Value;
            Sprite sprite = SpriteList[SpriteName];

            Tile newTile = new Tile(X * TileSize, Y * TileSize, Tileset, sprite.X,
sprite.Y, level);

            var Te = Tile.Elements();
            foreach(var Tee in Te)
            {
                if (Tee.Name == "trigger")
                {
                    newTile.AddTrigger(Tee.Value);
                }
            }
            newTile.Collides = sprite.Collides;

            Levellist.Add(newTile);
            X++;
            if (X >= RowLength)
            {
                Y++;
                X = 0;
            }
        }

        return Levellist;
    }

    /// <summary>
    /// Loads the sprite set.
    /// </summary>
    /// <returns>
    /// The sprite set.
    /// </returns>
    /// <param name='spriteSet'>
    /// Sprite set.
    /// </param>
    public static Dictionary<string, Sprite> LoadSpriteSet (string spriteSet)
    {
        Dictionary<string, Sprite> SpriteList = new Dictionary<string, Sprite>();

        XDocument TileSet = XDocument.Load("Content/Tilesets/" + spriteSet +
".xml");
        XElement TileSetDeclaration = TileSet.Element("tileset");
        int TileSize =
Int32.Parse(TileSetDeclaration.Attribute("tilesize").Value);

        IEnumerable<XElement> Sprites = TileSetDeclaration.Elements("sprite");

```

```

        foreach (XElement Sprite in Sprites)
        {
            int x = Int32.Parse(Sprite.Element("x").Value) * TileSize;
            int y = Int32.Parse(Sprite.Element("y").Value) * TileSize;

            string name = Sprite.Element("name").Value;
            bool collides = Boolean.Parse(Sprite.Element("collides").Value);

            Sprite newSprite = new Sprite(x, y, name, collides);
            SpriteList.Add(name, newSprite);
        }

        return SpriteList;
    }
}

```

Resource Manager

```

using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;

namespace AthenaEngine.Framework.Systems
{
    /// <summary>
    /// The ResourceManager class manages resources on behalf of the game.
    /// </summary>
    /// <typeparam name="T">The type of resource to manage</typeparam>
    public class ResourceManager<T>
    {
        private Game game;
        private Dictionary<String, T> Resources = new Dictionary<String, T>();

        /// <summary>
        /// Creates a new ResourceManager to manage resources for a game.
        /// </summary>
        /// <param name="game">The game to manage resources for.</param>
        public ResourceManager(Game game)
        {
            this.game = game;
        }

        /// <summary>
        /// Add a new resource to the resources list maintained by the
        ResourceManager.
        /// </summary>
        /// <param name="key">The key to associate the resource with</param>
        /// <param name="resource">The resource associated with the key</param>
        /// <returns></returns>
        public T Add(string key, T resource)
        {
            this.Resources.Add(key, resource);
            return resource;
        }
    }
}

```

```

    }

    /// <summary>
    /// Gets the resource stored associated with a key.
    /// </summary>
    /// <param name="key">The key to find the resource associated with.</param>
    /// <returns></returns>
    public T Get(string key)
    {
        return this.Resources[key];
    }
}
}

```

Triggers

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace AthenaEngine.Framework.Systems
{
    /// <summary>
    /// Triggers.
    /// </summary>
    class Triggers
    {
        /// <summary>
        /// This was used to test
        /// </summary>
        public static void test ()
        {
            System.Console.WriteLine("You stepped on a tile.");
        }

        /// <summary>
        /// Random encounter test
        /// </summary>
        public static void encounter()
        {
            Random Rng = new Random();
            int encounter = Rng.Next(1,20);

            if((encounter %4 == 0))
            {
                Console.WriteLine("Random Encounter!");
            }
        }
    }
}

```

UI

```

using System;

```

```

using System.Collections.Generic;
using System.Linq;
using System.Text;

using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;

using AthenaEngine.Framework;
using AthenaEngine.Framework.Gameplay;
using AthenaEngine.Framework.Interfaces;
using AthenaEngine.Framework.Primatives;
using AthenaEngine.Framework.Systems;

namespace AthenaEngine.Framework.UI
{
    /// <summary>
    /// UI
    /// </summary>
    public class UI
    {
        SpriteBatch spriteBatch;
        ResourceManager<Texture2D> textureManager;
        ResourceManager<SpriteFont> fontManager;
        public Level level;
        /*
        public UIButton(Vector2 position, Vector2 size, SpriteBatch spriteBatch,
Texture2D texture) : base(position, size, spriteBatch, texture)
        {
            this.SpriteColor = Color.White;
            this.SpriteTexture = texture;
            this.SpriteRectangle = new Rectangle((int)position.X, (int)position.Y,
(int)size.X, (int)size.Y);
            this.SpriteController = spriteBatch;
        }
        */

        /// <summary>
        /// Initializes a new instance of the <see
cref="AthenaEngine.Framework.UI.UI"/> class.
        /// </summary>
        /// <param name='spriteBatch'>
        /// Sprite batch.
        /// </param>
        /// <param name='textureManager'>
        /// Texture manager.
        /// </param>
        /// <param name='fontManager'>
        /// Font manager.
        /// </param>
        public UI(SpriteBatch spriteBatch, ResourceManager<Texture2D> textureManager,
ResourceManager<SpriteFont> fontManager)
        {
            this.SpriteBatch = spriteBatch;
            this.TextureManager = textureManager;
            this.FontManager = fontManager;
        }
    }
}

```



```

        /// <summary>
        /// The buttons.
        /// </summary>
        private List<UIButton> Buttons = new List<UIButton>();

        /// <summary>
        /// Adds the button.
        /// </summary>
        /// <param name='position'>
        /// Position.
        /// </param>
        /// <param name='label'>
        /// Label.
        /// </param>
        public void AddButton(Vector2 position, string label)
        {
            // Buttons.Add(new UIButton(position, new Vector2(100, 40), SpriteBatch,
            TextureManager.Get("blank")));
            Buttons.Add(new UIButton(new Rectangle((int)position.X, (int)position.Y,
            100, 40), this.SpriteBatch, TextureManager.Get("blank"), Color.Orange, label,
            FontManager.Get("SpriteFont1")));
        }

        /// <summary>
        /// Draw this instance.
        /// </summary>
        public void Draw ()
        {
            foreach (UIButton btn in Buttons)
            {
                btn.Draw();
            }
        }
    }
}

```

UI Button

```

using System;
using System.Collections.Generic;
using System.Linq;

using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;

using AthenaEngine.Framework;
using AthenaEngine.Framework.Gameplay;
using AthenaEngine.Framework.Interfaces;
using AthenaEngine.Framework.Primitives;
using AthenaEngine.Framework.Systems;

namespace AthenaEngine.Framework.UI
{

```

```

    /// <summary>
    /// User interface button.
    /// </summary>
public class UIButton
{
    private SpriteBatch Batch;
    private Texture2D Texture;
    private SpriteFont Font;
    private Color Color;
    private string Label;
    public Level Level;

    private Rectangle SpriteRect;

    /// <summary>
    /// Initializes a new instance of the <see
    cref="AthenaEngine.Framework.UI.UIButton"/> class.
    /// </summary>
    /// <param name='rectangle'>
    /// Rectangle.
    /// </param>
    /// <param name='spriteBatch'>
    /// Sprite batch.
    /// </param>
    /// <param name='texture'>
    /// Texture.
    /// </param>
    /// <param name='color'>
    /// Color.
    /// </param>
    /// <param name='label'>
    /// Label.
    /// </param>
    /// <param name='font'>
    /// Font.
    /// </param>
    public UIButton (Rectangle rectangle, SpriteBatch spriteBatch, Texture2D
    texture, Color color, string label, SpriteFont font)
    {
        this.SpriteRect = rectangle;
        this.Batch = spriteBatch;
        this.Texture = texture;
        this.Color = color;
        this.Label = label;
        this.Font = font;
    }

    /// <summary>
    /// Draw this instance.
    /// </summary>
    public void Draw ()
    {
        // DrawableEntity Base = new DrawableEntity(new Vector2(SpriteRect.X,
    SpriteRect.Y), new Vector2(SpriteRect.Width, SpriteRect.Height), Batch, Texture,
    Level);
        // DrawableEntity Stroke = new DrawableEntity(new Vector2(SpriteRect.X -
    2, SpriteRect.Y - 2), new Vector2(SpriteRect.Width + 4, SpriteRect.Height + 4), Batch,
    Texture, Level);
        // Stroke.SpriteColor = Color.Black;
        // Base.SpriteColor = this.Color;
        // Stroke.Draw();
        // Base.Draw();
    }
}

```

```

        // int x = (int)SpriteRect.X + Font.MeasureString(Label).X / 2

        int X = SpriteRect.X + (SpriteRect.Width - (int)
Font.MeasureString(Label).X) / 2;
        int Y = SpriteRect.Y;
        Batch.DrawString(Font, Label, new Vector2(X, Y), Color.Black);
    }
}
}

```

Camera 2D

```

using System;
using System.Collections.Generic;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;

using AthenaEngine.Framework.Interfaces;

namespace AthenaEngine.Framework
{
    /// <summary>
    /// Camera2D
    /// </summary>
    public class Camera2D : GameComponent
    {
        private Vector2 _position;
        protected float _viewportHeight;
        protected float _viewportWidth;

        /// <summary>
        /// Initializes a new instance of the <see
cref="AthenaEngine.Framework.Camera2D"/> class.
        /// </summary>
        /// <param name='game'>
        /// Game.
        /// </param>
        public Camera2D (Game game) : base (game)
        {
        }

        /// <summary>
        /// Gets or sets the position.
        /// </summary>
        /// <value>
        /// The position.
        /// </value>
        public Vector2 Position {
            get { return _position; }
            set { _position = value; }
        }

        public float Rotation { get; set; }
    }
}

```

```

        public Vector2 Origin { get; set; }
        public float Scale { get; set; }
        public Vector2 ScreenCenter { get; protected set; }
        public Matrix Transform { get; set; }
        public IFocusable Focus { get; set; }
        public float MoveSpeed { get; set; }

        /// <summary>
        /// Called when the GameComponent needs to be initialized.
        /// </summary>
        ///
        public override void Initialize()
        {
            _viewportWidth = Game.GraphicsDevice.Viewport.Width;
            _viewportHeight = Game.GraphicsDevice.Viewport.Height;

            ScreenCenter = new Vector2(_viewportWidth/2, _viewportHeight/2);
            Scale = 1;
            MoveSpeed = 1.25f;

            base.Initialize();
        }

        /// <summary>
        /// Update the specified gameTime.
        /// </summary>
        /// <param name='gameTime'>
        /// Game time.
        /// </param>
        public override void Update(GameTime gameTime)
        {
            // Create the Transform used by any
            // spritebatch process
            Transform = Matrix.Identity*
                Matrix.CreateTranslation(-Position.X, -Position.Y, 0)*
                Matrix.CreateRotationZ(Rotation)*
                Matrix.CreateTranslation(Origin.X, Origin.Y, 0)*
                Matrix.CreateScale(new Vector3(Scale, Scale, Scale));

            Origin = ScreenCenter / Scale;

            // Move the Camera to the position that it needs to go
            var delta = (float) gameTime.ElapsedGameTime.TotalSeconds;

            _position.X += (Focus.Position.X - Position.X) * MoveSpeed * delta;
            _position.Y += (Focus.Position.Y - Position.Y) * MoveSpeed * delta;

            base.Update(gameTime);
        }

        /// <summary>
        /// Determines whether the target is in view given the specified position.
        /// This can be used to increase performance by not drawing objects
        /// directly in the viewport
        /// </summary>
        /// <param name="position">The position.</param>
        /// <param name="texture">The texture.</param>
        /// <returns>
        /// <c>true</c> if [is in view] [the specified position]; otherwise,
        <c>false</c>.
        /// </returns>
        public bool IsInView(Vector2 position, Texture2D texture)

```

```

    {
        // If the object is not within the horizontal bounds of the screen
        if ( (position.X + texture.Width) < (Position.X - Origin.X) || (position.X) >
(Position.X + Origin.X) )
            return false;

        // If the object is not within the vertical bounds of the screen
        if ((position.Y + texture.Height) < (Position.Y - Origin.Y) || (position.Y) >
(Position.Y + Origin.Y))
            return false;

        // In View
        return true;
    }
}

```

Bounding Box 2D Test

```

using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using System.Collections.Generic;
using System.Linq;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;

using AthenaEngine.Framework.Primitives;

namespace AthenaTest.Engine_Tests.Primitives_Testing
{
    [TestClass]
    public class BoundingBox2DTest
    {
        [TestMethod]
        public void Equality_WithSameRectangle_IsEqual()
        {
            Vector2 FirstBoundingBox2D_XY = new Vector2(0, 0);
            Vector2 FirstBoundingBox2D_WidthHeight = new Vector2(100, 100);

            BoundingBox2D FirstBoundingBox2D = new
BoundingBox2D(FirstBoundingBox2D_XY, FirstBoundingBox2D_WidthHeight);
            BoundingBox2D SecondBoundingBox2D = FirstBoundingBox2D;

            bool BoxesEqual = FirstBoundingBox2D.Equals(SecondBoundingBox2D);

            Assert.AreEqual<bool>(BoxesEqual, true);
        }

        [TestMethod]
        public void Equality_WithIdenticalRectangle_IsEqual()
        {
            Vector2 FirstBoundingBox2D_XY = new Vector2(0, 0);
            Vector2 FirstBoundingBox2D_WidthHeight = new Vector2(100, 100);

```

```

        BoundingBox2D FirstBoundingBox2D = new
BoundingBox2D(FirstBoundingBox2D_XY, FirstBoundingBox2D_WidthHeight);
        BoundingBox2D SecondBoundingBox2D = new
BoundingBox2D(FirstBoundingBox2D_XY, FirstBoundingBox2D_WidthHeight);

        bool BoxesEqual = FirstBoundingBox2D.Equals(SecondBoundingBox2D);

        Assert.AreEqual<bool>(BoxesEqual, true);
    }

    [TestMethod]
    public void Equality_WithDifferentRectangle_IsNotEqual()
    {
        Vector2 FirstBoundingBox2D_XY = new Vector2(0, 0);
        Vector2 FirstBoundingBox2D_WidthHeight = new Vector2(100, 100);

        Vector2 SecondBoundingBox2D_XY = new Vector2(32, 15);
        Vector2 SecondBoundingBox2D_WidthHeight = new Vector2(81, 542);

        BoundingBox2D FirstBoundingBox2D = new
BoundingBox2D(FirstBoundingBox2D_XY, FirstBoundingBox2D_WidthHeight);
        BoundingBox2D SecondBoundingBox2D = new
BoundingBox2D(SecondBoundingBox2D_XY, SecondBoundingBox2D_WidthHeight);

        bool BoxesEqual = FirstBoundingBox2D.Equals(SecondBoundingBox2D);

        Assert.AreEqual<bool>(BoxesEqual, false);
    }

    [TestMethod]
    public void Equality_WithItself_IsEqual()
    {
        Vector2 FirstBoundingBox2D_XY = new Vector2(0, 0);
        Vector2 FirstBoundingBox2D_WidthHeight = new Vector2(100, 100);

        BoundingBox2D FirstBoundingBox2D = new
BoundingBox2D(FirstBoundingBox2D_XY, FirstBoundingBox2D_WidthHeight);

        bool BoxesEqual = FirstBoundingBox2D.Equals(FirstBoundingBox2D);

        Assert.AreEqual<bool>(BoxesEqual, true);
    }

    [TestMethod]
    public void Collisions_WithItself_IsTrue()
    {
        Vector2 FirstBoundingBox2D_XY = new Vector2(0, 0);
        Vector2 FirstBoundingBox2D_WidthHeight = new Vector2(100, 100);

        BoundingBox2D FirstBoundingBox2D = new
BoundingBox2D(FirstBoundingBox2D_XY, FirstBoundingBox2D_WidthHeight);

        bool BoxesCollide = FirstBoundingBox2D.CollidesWith(FirstBoundingBox2D);

        Assert.AreEqual<bool>(BoxesCollide, true);
    }

    [TestMethod]
    public void Collisions_WithTotallyDifferentRectangle_IsNotTrue()
    {
        Vector2 FirstBoundingBox2D_XY = new Vector2(0, 0);
        Vector2 FirstBoundingBox2D_WidthHeight = new Vector2(100, 100);

```

```

        Vector2 SecondBoundingBox2D_XY = new Vector2(151, 214);
        Vector2 SecondBoundingBox2D_WidthHeight = new Vector2(81, 542);

        BoundingBox2D FirstBoundingBox2D = new
BoundingBox2D(FirstBoundingBox2D_XY, FirstBoundingBox2D_WidthHeight);
        BoundingBox2D SecondBoundingBox2D = new
BoundingBox2D(SecondBoundingBox2D_XY, SecondBoundingBox2D_WidthHeight);

        bool BoxesCollide = FirstBoundingBox2D.CollidesWith(SecondBoundingBox2D);

        Assert.AreEqual<bool>(BoxesCollide, false);
    }

    [TestMethod]
    public void Collisions_WithEnvelopedRectangle_IsTrue()
    {
        Vector2 FirstBoundingBox2D_XY = new Vector2(0, 0);
        Vector2 FirstBoundingBox2D_WidthHeight = new Vector2(100, 100);

        Vector2 SecondBoundingBox2D_XY = new Vector2(25, 25);
        Vector2 SecondBoundingBox2D_WidthHeight = new Vector2(25, 25);

        BoundingBox2D FirstBoundingBox2D = new
BoundingBox2D(FirstBoundingBox2D_XY, FirstBoundingBox2D_WidthHeight);
        BoundingBox2D SecondBoundingBox2D = new
BoundingBox2D(SecondBoundingBox2D_XY, SecondBoundingBox2D_WidthHeight);

        bool BoxesCollide = FirstBoundingBox2D.CollidesWith(SecondBoundingBox2D);

        Assert.AreEqual<bool>(BoxesCollide, true);
    }

    [TestMethod]
    public void Collisions_WithRectangleOnLeft_IsFalse()
    {
        Vector2 FirstBoundingBox2D_XY = new Vector2(10, 10);
        Vector2 FirstBoundingBox2D_WidthHeight = new Vector2(10, 10);

        Vector2 SecondBoundingBox2D_XY = new Vector2(0, 10);
        Vector2 SecondBoundingBox2D_WidthHeight = new Vector2(10, 10);

        BoundingBox2D FirstBoundingBox2D = new
BoundingBox2D(FirstBoundingBox2D_XY, FirstBoundingBox2D_WidthHeight);
        BoundingBox2D SecondBoundingBox2D = new
BoundingBox2D(SecondBoundingBox2D_XY, SecondBoundingBox2D_WidthHeight);

        bool BoxesCollide = FirstBoundingBox2D.CollidesWith(SecondBoundingBox2D);

        Assert.AreEqual<bool>(BoxesCollide, false);
    }

    [TestMethod]
    public void Collisions_WithRectangleOnRight_IsFalse()
    {
        Vector2 FirstBoundingBox2D_XY = new Vector2(10, 10);
        Vector2 FirstBoundingBox2D_WidthHeight = new Vector2(10, 10);

        Vector2 SecondBoundingBox2D_XY = new Vector2(20, 10);
        Vector2 SecondBoundingBox2D_WidthHeight = new Vector2(10, 10);

```

```

        BoundingBox2D FirstBoundingBox2D = new
BoundingBox2D(FirstBoundingBox2D_XY, FirstBoundingBox2D_WidthHeight);
        BoundingBox2D SecondBoundingBox2D = new
BoundingBox2D(SecondBoundingBox2D_XY, SecondBoundingBox2D_WidthHeight);

        bool BoxesCollide = FirstBoundingBox2D.CollidesWith(SecondBoundingBox2D);

        Assert.AreEqual<bool>(BoxesCollide, false);
    }

    [TestMethod]
    public void Collisions_WithRectangleOnTop_IsFalse()
    {
        Vector2 FirstBoundingBox2D_XY = new Vector2(10, 10);
        Vector2 FirstBoundingBox2D_WidthHeight = new Vector2(10, 10);

        Vector2 SecondBoundingBox2D_XY = new Vector2(10, 0);
        Vector2 SecondBoundingBox2D_WidthHeight = new Vector2(10, 10);

        BoundingBox2D FirstBoundingBox2D = new
BoundingBox2D(FirstBoundingBox2D_XY, FirstBoundingBox2D_WidthHeight);
        BoundingBox2D SecondBoundingBox2D = new
BoundingBox2D(SecondBoundingBox2D_XY, SecondBoundingBox2D_WidthHeight);

        bool BoxesCollide = FirstBoundingBox2D.CollidesWith(SecondBoundingBox2D);

        Assert.AreEqual<bool>(BoxesCollide, false);
    }

    [TestMethod]
    public void Collisions_WithRectangleOnBottom_IsFalse()
    {
        Vector2 FirstBoundingBox2D_XY = new Vector2(10, 10);
        Vector2 FirstBoundingBox2D_WidthHeight = new Vector2(10, 10);

        Vector2 SecondBoundingBox2D_XY = new Vector2(10, 20);
        Vector2 SecondBoundingBox2D_WidthHeight = new Vector2(10, 10);

        BoundingBox2D FirstBoundingBox2D = new
BoundingBox2D(FirstBoundingBox2D_XY, FirstBoundingBox2D_WidthHeight);
        BoundingBox2D SecondBoundingBox2D = new
BoundingBox2D(SecondBoundingBox2D_XY, SecondBoundingBox2D_WidthHeight);

        bool BoxesCollide = FirstBoundingBox2D.CollidesWith(SecondBoundingBox2D);

        Assert.AreEqual<bool>(BoxesCollide, false);
    }

    [TestMethod]
    public void Collisions_WithRectangleCollidingLeft_IsTrue()
    {
        Vector2 FirstBoundingBox2D_XY = new Vector2(10, 10);
        Vector2 FirstBoundingBox2D_WidthHeight = new Vector2(10, 10);

        Vector2 SecondBoundingBox2D_XY = new Vector2(1, 10);
        Vector2 SecondBoundingBox2D_WidthHeight = new Vector2(10, 10);

        BoundingBox2D FirstBoundingBox2D = new
BoundingBox2D(FirstBoundingBox2D_XY, FirstBoundingBox2D_WidthHeight);
        BoundingBox2D SecondBoundingBox2D = new
BoundingBox2D(SecondBoundingBox2D_XY, SecondBoundingBox2D_WidthHeight);

```



```

        bool BoxesCollide = FirstBoundingBox2D.CollidesWith(SecondBoundingBox2D);

        Assert.AreEqual<bool>(BoxesCollide, true);
    }

    [TestMethod]
    public void Collisions_WithRectangleCollidingRight_IsTrue()
    {
        Vector2 FirstBoundingBox2D_XY = new Vector2(10, 10);
        Vector2 FirstBoundingBox2D_WidthHeight = new Vector2(10, 10);

        Vector2 SecondBoundingBox2D_XY = new Vector2(19, 10);
        Vector2 SecondBoundingBox2D_WidthHeight = new Vector2(10, 10);

        BoundingBox2D FirstBoundingBox2D = new
BoundingBox2D(FirstBoundingBox2D_XY, FirstBoundingBox2D_WidthHeight);
        BoundingBox2D SecondBoundingBox2D = new
BoundingBox2D(SecondBoundingBox2D_XY, SecondBoundingBox2D_WidthHeight);

        bool BoxesCollide = FirstBoundingBox2D.CollidesWith(SecondBoundingBox2D);

        Assert.AreEqual<bool>(BoxesCollide, true);
    }

    [TestMethod]
    public void Collisions_WithRectangleCollidingTop_IsTrue()
    {
        Vector2 FirstBoundingBox2D_XY = new Vector2(10, 10);
        Vector2 FirstBoundingBox2D_WidthHeight = new Vector2(10, 10);

        Vector2 SecondBoundingBox2D_XY = new Vector2(10, 1);
        Vector2 SecondBoundingBox2D_WidthHeight = new Vector2(10, 10);

        BoundingBox2D FirstBoundingBox2D = new
BoundingBox2D(FirstBoundingBox2D_XY, FirstBoundingBox2D_WidthHeight);
        BoundingBox2D SecondBoundingBox2D = new
BoundingBox2D(SecondBoundingBox2D_XY, SecondBoundingBox2D_WidthHeight);

        bool BoxesCollide = FirstBoundingBox2D.CollidesWith(SecondBoundingBox2D);

        Assert.AreEqual<bool>(BoxesCollide, true);
    }

    [TestMethod]
    public void Collisions_WithRectangleCollidingBottom_IsTrue()
    {
        Vector2 FirstBoundingBox2D_XY = new Vector2(10, 10);
        Vector2 FirstBoundingBox2D_WidthHeight = new Vector2(10, 10);

        Vector2 SecondBoundingBox2D_XY = new Vector2(10, 19);
        Vector2 SecondBoundingBox2D_WidthHeight = new Vector2(10, 10);

        BoundingBox2D FirstBoundingBox2D = new
BoundingBox2D(FirstBoundingBox2D_XY, FirstBoundingBox2D_WidthHeight);
        BoundingBox2D SecondBoundingBox2D = new
BoundingBox2D(SecondBoundingBox2D_XY, SecondBoundingBox2D_WidthHeight);

        bool BoxesCollide = FirstBoundingBox2D.CollidesWith(SecondBoundingBox2D);

        Assert.AreEqual<bool>(BoxesCollide, true);
    }
}

```


9 Appendix 9 - Code Documentation

Athena

Generated by Doxygen 1.8.3.1

Tue May 6 2014 12:11:10

Contents

1	Namespace Index	1
1.1	Packages	1
2	Hierarchical Index	3
2.1	Class Hierarchy	3
3	Class Index	5
3.1	Class List	5
4	File Index	7
4.1	File List	7
5	Namespace Documentation	9
5.1	Package Athena	9
5.2	Package AthenaEngine	9
5.3	Package AthenaEngine.Framework	9
5.4	Package AthenaEngine.Framework.Gameplay	10
5.5	Package AthenaEngine.Framework.Gameplay.RPG	10
5.6	Package AthenaEngine.Framework.Interfaces	10
5.7	Package AthenaEngine.Framework.Primitives	10
5.8	Package AthenaEngine.Framework.Systems	11
5.9	Package AthenaEngine.Framework.UI	11
5.10	Package AthenaTest	11
5.11	Package AthenaTest.Engine_Tests	11
5.12	Package AthenaTest.Engine_Tests.Primitives_Testing	11
6	Class Documentation	13
6.1	AthenaEngine.AthenaEngine Class Reference	13
6.2	AthenaEngine.Framework.Primitives.BoundingBox2D Class Reference	13
6.2.1	Detailed Description	13
6.2.2	Constructor & Destructor Documentation	14
6.2.2.1	BoundingBox2D	14
6.2.3	Member Function Documentation	14

6.2.3.1	CollidesWith	14
6.2.3.2	Equals	14
6.2.4	Property Documentation	14
6.2.4.1	Bounds	14
6.3	AthenaTest.Engine_Tests.Primitives_Testing.BoundingBox2DTest Class Reference	14
6.3.1	Member Function Documentation	15
6.3.1.1	Collisions_WithEnvelopedRectangle_IsTrue	15
6.3.1.2	Collisions_WithItself_IsTrue	15
6.3.1.3	Collisions_WithRectangleCollidingBottom_IsTrue	15
6.3.1.4	Collisions_WithRectangleCollidingLeft_IsTrue	15
6.3.1.5	Collisions_WithRectangleCollidingRight_IsTrue	15
6.3.1.6	Collisions_WithRectangleCollidingTop_IsTrue	15
6.3.1.7	Collisions_WithRectangleOnBottom_IsFalse	15
6.3.1.8	Collisions_WithRectangleOnLeft_IsFalse	15
6.3.1.9	Collisions_WithRectangleOnRight_IsFalse	15
6.3.1.10	Collisions_WithRectangleOnTop_IsFalse	15
6.3.1.11	Collisions_WithTotallyDifferentRectangle_IsNotTrue	15
6.3.1.12	Equality_WithDifferentRectangle_IsNotEqual	15
6.3.1.13	Equality_WithIdenticalRectangle_IsEqual	15
6.3.1.14	Equality_WithItself_IsEqual	15
6.3.1.15	Equality_WithSameRectangle_IsEqual	15
6.4	AthenaEngine.Framework.Camera2D Class Reference	16
6.4.1	Detailed Description	16
6.4.2	Constructor & Destructor Documentation	16
6.4.2.1	Camera2D	16
6.4.3	Member Function Documentation	17
6.4.3.1	Initialize	17
6.4.3.2	IsInView	17
6.4.3.3	Update	17
6.4.4	Member Data Documentation	17
6.4.4.1	_viewportHeight	17
6.4.4.2	_viewportWidth	17
6.4.5	Property Documentation	17
6.4.5.1	Focus	17
6.4.5.2	MoveSpeed	17
6.4.5.3	Origin	17
6.4.5.4	Position	17
6.4.5.5	Rotation	17
6.4.5.6	Scale	18
6.4.5.7	ScreenCenter	18

6.4.5.8	Transform	18
6.5	AthenaEngine.Framework.Gameplay.Character Class Reference	18
6.5.1	Detailed Description	18
6.5.2	Constructor & Destructor Documentation	18
6.5.2.1	Character	18
6.6	AthenaEngine.Framework.Primitives.CollidableEntity Class Reference	19
6.6.1	Detailed Description	19
6.6.2	Constructor & Destructor Documentation	19
6.6.2.1	CollidableEntity	19
6.6.3	Member Function Documentation	20
6.6.3.1	CanMove	20
6.6.3.2	CollidesWith	20
6.6.3.3	Move	20
6.6.4	Member Data Documentation	20
6.6.4.1	Level	20
6.6.5	Property Documentation	20
6.6.5.1	Bounds	20
6.7	AthenaEngine.Framework.Primitives.Directions Class Reference	20
6.7.1	Member Data Documentation	21
6.7.1.1	DOWN	21
6.7.1.2	LEFT	21
6.7.1.3	RIGHT	21
6.7.1.4	UP	21
6.8	AthenaEngine.Framework.Primitives.DrawableEntity Class Reference	21
6.8.1	Detailed Description	22
6.8.2	Constructor & Destructor Documentation	22
6.8.2.1	DrawableEntity	22
6.8.3	Member Function Documentation	22
6.8.3.1	Draw	22
6.8.4	Member Data Documentation	22
6.8.4.1	level	22
6.8.4.2	SpriteColor	22
6.8.4.3	SpriteController	22
6.8.4.4	SpriteSheet	22
6.8.4.5	SpriteSource	22
6.8.5	Property Documentation	22
6.8.5.1	Position	22
6.9	AthenaEngine.Framework.Primitives.Entity Class Reference	23
6.9.1	Detailed Description	23
6.9.2	Member Function Documentation	23

6.9.2.1	Move	23
6.9.3	Member Data Documentation	23
6.9.3.1	Position	23
6.9.3.2	Size	23
6.9.4	Property Documentation	23
6.9.4.1	Height	23
6.9.4.2	Rectangle	23
6.9.4.3	Width	24
6.9.4.4	X	24
6.9.4.5	Y	24
6.10	Athena.Game1 Class Reference	24
6.10.1	Detailed Description	24
6.10.2	Constructor & Destructor Documentation	25
6.10.2.1	Game1	25
6.10.3	Member Function Documentation	25
6.10.3.1	Draw	25
6.10.3.2	Initialize	25
6.10.3.3	LoadContent	25
6.10.3.4	UnloadContent	25
6.10.3.5	Update	25
6.10.4	Member Data Documentation	25
6.10.4.1	font	25
6.11	AthenaEngine.Framework.Interfaces.ICollidable< T > Interface Template Reference	25
6.11.1	Detailed Description	26
6.11.2	Member Function Documentation	26
6.11.2.1	CollidesWith	26
6.12	AthenaEngine.Framework.Interfaces.IDrawable Interface Reference	26
6.12.1	Detailed Description	26
6.12.2	Member Function Documentation	26
6.12.2.1	Draw	26
6.13	AthenaEngine.Framework.Interfaces.IFocusable Interface Reference	27
6.13.1	Detailed Description	27
6.13.2	Property Documentation	27
6.13.2.1	Position	27
6.14	AthenaEngine.Framework.Interfaces.IMoveable Interface Reference	27
6.14.1	Member Function Documentation	27
6.14.1.1	CanMove	27
6.14.1.2	Move	27
6.15	AthenaEngine.Framework.Gameplay.RPG.Inventory Class Reference	28
6.15.1	Detailed Description	28

6.15.2 Member Function Documentation	28
6.15.2.1 Add	28
6.16 AthenaEngine.Framework.Gameplay.RPG.Item Class Reference	28
6.16.1 Detailed Description	29
6.16.2 Constructor & Destructor Documentation	29
6.16.2.1 Item	29
6.16.3 Property Documentation	29
6.16.3.1 Name	29
6.17 AthenaEngine.Framework.Gameplay.RPG.ItemInstance Class Reference	29
6.17.1 Detailed Description	29
6.17.2 Constructor & Destructor Documentation	29
6.17.2.1 ItemInstance	29
6.18 AthenaEngine.Framework.Gameplay.Level Class Reference	30
6.18.1 Detailed Description	30
6.18.2 Constructor & Destructor Documentation	30
6.18.2.1 Level	30
6.18.3 Member Function Documentation	30
6.18.3.1 Draw	30
6.18.4 Member Data Documentation	30
6.18.4.1 TileList	31
6.19 AthenaEngine.Framework.Systems.ResourceManager< T > Class Template Reference	31
6.19.1 Detailed Description	31
6.19.2 Constructor & Destructor Documentation	31
6.19.2.1 ResourceManager	31
6.19.3 Member Function Documentation	31
6.19.3.1 Add	31
6.19.3.2 Get	32
6.20 AthenaEngine.Framework.Systems.LevelLoaderXml.Sprite Class Reference	32
6.20.1 Constructor & Destructor Documentation	32
6.20.1.1 Sprite	32
6.20.2 Member Data Documentation	32
6.20.2.1 Collides	32
6.20.2.2 Name	32
6.20.2.3 X	32
6.20.2.4 Y	32
6.21 AthenaEngine.Framework.Gameplay.Tile Class Reference	32
6.21.1 Detailed Description	33
6.21.2 Constructor & Destructor Documentation	33
6.21.2.1 Tile	33
6.21.3 Member Function Documentation	33

6.21.3.1	AddTrigger	33
6.21.3.2	Draw	34
6.21.3.3	FireTrigger	34
6.21.3.4	MakeDrawable	34
6.21.4	Member Data Documentation	34
6.21.4.1	Collides	34
6.21.4.2	HasTrigger	34
6.21.4.3	Sprite	34
6.22	AthenaEngine.Framework.Systems.Triggers Class Reference	34
6.22.1	Detailed Description	35
6.22.2	Member Function Documentation	35
6.22.2.1	encounter	35
6.22.2.2	test	35
6.23	AthenaEngine.Framework.UI.UI Class Reference	35
6.23.1	Detailed Description	35
6.23.2	Constructor & Destructor Documentation	35
6.23.2.1	UI	35
6.23.3	Member Function Documentation	36
6.23.3.1	AddButton	36
6.23.3.2	Draw	36
6.23.4	Member Data Documentation	36
6.23.4.1	Level	36
6.24	AthenaEngine.Framework.UI.UIButton Class Reference	36
6.24.1	Detailed Description	36
6.24.2	Constructor & Destructor Documentation	36
6.24.2.1	UIButton	36
6.24.3	Member Function Documentation	37
6.24.3.1	Draw	37
6.24.4	Member Data Documentation	37
6.24.4.1	Level	37
7	File Documentation	39
7.1	Athena/Athena/Athena/Debug.cs File Reference	39
7.2	Athena/Athena/Athena/Game1.cs File Reference	39
7.3	Athena/Athena/Athena/Program.cs File Reference	39
7.4	Athena/Athena/Athena/Properties/AssemblyInfo.cs File Reference	39
7.5	Athena/Athena/AthenaEngine/Properties/AssemblyInfo.cs File Reference	39
7.6	Athena/Athena/AthenaTest/Properties/AssemblyInfo.cs File Reference	39
7.7	Athena/Athena/AthenaEngine/AthenaEngine.cs File Reference	40
7.8	Athena/Athena/AthenaEngine/Framework/Camera2D.cs File Reference	40

7.9	Athena/Athena/AthenaEngine/Framework/Gameplay/Character.cs File Reference	40
7.10	Athena/Athena/AthenaEngine/Framework/Gameplay/Level.cs File Reference	40
7.11	Athena/Athena/AthenaEngine/Framework/Gameplay/RPG/Inventory.cs File Reference	41
7.12	Athena/Athena/AthenaEngine/Framework/Gameplay/RPG/Item.cs File Reference	41
7.13	Athena/Athena/AthenaEngine/Framework/Gameplay/RPG/ItemInstance.cs File Reference	41
7.14	Athena/Athena/AthenaEngine/Framework/Gameplay/Tile.cs File Reference	41
7.15	Athena/Athena/AthenaEngine/Framework/Interfaces/ICollidable.cs File Reference	42
7.16	Athena/Athena/AthenaEngine/Framework/Interfaces/IDrawable.cs File Reference	42
7.17	Athena/Athena/AthenaEngine/Framework/Interfaces/IFocusable.cs File Reference	42
7.18	Athena/Athena/AthenaEngine/Framework/Interfaces/IMoveable.cs File Reference	42
7.19	Athena/Athena/AthenaEngine/Framework/Primitives/BoundingBox2D.cs File Reference	43
7.20	Athena/Athena/AthenaEngine/Framework/Primitives/CollidableEntity.cs File Reference	43
7.21	Athena/Athena/AthenaEngine/Framework/Primitives/Directions.cs File Reference	43
7.22	Athena/Athena/AthenaEngine/Framework/Primitives/DrawableEntity.cs File Reference	43
7.23	Athena/Athena/AthenaEngine/Framework/Primitives/Entity.cs File Reference	44
7.24	Athena/Athena/AthenaEngine/Framework/Systems/LevelLoaderXml.cs File Reference	44
7.25	Athena/Athena/AthenaEngine/Framework/Systems/ResourceManager.cs File Reference	44
7.26	Athena/Athena/AthenaEngine/Framework/Systems/Triggers.cs File Reference	44
7.27	Athena/Athena/AthenaEngine/Framework/UI/UI.cs File Reference	45
7.28	Athena/Athena/AthenaEngine/Framework/UI/UIButton.cs File Reference	45
7.29	Athena/Athena/AthenaEngine/obj/Debug/TemporaryGeneratedFile_036C0B5B-1481-4323-8D20-8F5-ADCB23D92.cs File Reference	45
7.30	Athena/Athena/AthenaTest/obj/Debug/TemporaryGeneratedFile_036C0B5B-1481-4323-8D20-8F5-ADCB23D92.cs File Reference	45
7.31	Athena/Athena/AthenaEngine/obj/Debug/TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs File Reference	45
7.32	Athena/Athena/AthenaTest/obj/Debug/TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs File Reference	45
7.33	Athena/Athena/AthenaEngine/obj/Debug/TemporaryGeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs File Reference	45
7.34	Athena/Athena/AthenaTest/obj/Debug/TemporaryGeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs File Reference	45
7.35	Athena/Athena/AthenaTest/Engine Tests/Primitives Testing/BoundingBox2DTest.cs File Reference	45

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

Athena	9
AthenaEngine	9
AthenaEngine.Framework	9
AthenaEngine.Framework.Gameplay	10
AthenaEngine.Framework.Gameplay.RPG	10
AthenaEngine.Framework.Interfaces	10
AthenaEngine.Framework.Primitives	10
AthenaEngine.Framework.Systems	11
AthenaEngine.Framework.UI	11
AthenaTest	11
AthenaTest.Engine_Tests	11
AthenaTest.Engine_Tests.Primitives_Testing	11

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

AthenaEngine.AthenaEngine	13
AthenaTest.Engine_Tests.Primatives_Testing.BoundingBox2DTest	14
AthenaEngine.Framework.Primatives.Directions	20
AthenaEngine.Framework.Primatives.Entity	23
AthenaEngine.Framework.Primatives.BoundingBox2D	13
AthenaEngine.Framework.Primatives.CollidableEntity	19
AthenaEngine.Framework.Gameplay.Tile	32
AthenaEngine.Framework.Primatives.DrawableEntity	21
AthenaEngine.Framework.Gameplay.Character	18
Game	
Athena.Game1	24
GameComponent	
AthenaEngine.Framework.Camera2D	16
AthenaEngine.Framework.Interfaces.ICollidable< T >	25
AthenaEngine.Framework.Interfaces.IDrawable	26
AthenaEngine.Framework.Primatives.DrawableEntity	21
IEquatable< BoundingBox2D >	
AthenaEngine.Framework.Primatives.BoundingBox2D	13
AthenaEngine.Framework.Interfaces.IFocusable	27
AthenaEngine.Framework.Gameplay.Character	18
AthenaEngine.Framework.Primatives.DrawableEntity	21
AthenaEngine.Framework.Interfaces.IMoveable	27
AthenaEngine.Framework.Gameplay.RPG.Inventory	28
AthenaEngine.Framework.Gameplay.RPG.Item	28
AthenaEngine.Framework.Gameplay.RPG.ItemInstance	29
AthenaEngine.Framework.Gameplay.Level	30
AthenaEngine.Framework.Systems.ResourceManager< T >	31
AthenaEngine.Framework.Systems.LevelLoaderXml.Sprite	32
AthenaEngine.Framework.Systems.Triggers	34
AthenaEngine.Framework.UI.UI	35
AthenaEngine.Framework.UI.UIButton	36

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

AthenaEngine.AthenaEngine	13
AthenaEngine.Framework.Primitives.BoundingBox2D	
BoundingBox2D is used for bounding boxes on 2D objects.	13
AthenaTest.Engine_Tests.Primitives_Testing.BoundingBox2DTest	14
AthenaEngine.Framework.Camera2D	
Camera2D	16
AthenaEngine.Framework.Gameplay.Character	
A character class holds important detail about each character such as their items, level, experience, skills, etc.	18
AthenaEngine.Framework.Primitives.CollidableEntity	
The Entity class is used to store objects that have positions.	19
AthenaEngine.Framework.Primitives.Directions	20
AthenaEngine.Framework.Primitives.DrawableEntity	
This is an entity which can be drawn.	21
AthenaEngine.Framework.Primitives.Entity	
The Entity class is the superclass for anything.	23
Athena.Game1	
This is the main type for your game	24
AthenaEngine.Framework.Interfaces.ICollidable< T >	
I collidable.	25
AthenaEngine.Framework.Interfaces.IDrawable	
I drawable.	26
AthenaEngine.Framework.Interfaces.IFocusable	
I focusable.	27
AthenaEngine.Framework.Interfaces.IMoveable	27
AthenaEngine.Framework.Gameplay.RPG.Inventory	
Inventory .	28
AthenaEngine.Framework.Gameplay.RPG.Item	
A gameplay item can be held by a character.	28
AthenaEngine.Framework.Gameplay.RPG.ItemInstance	
An item instance is a particular instance of an item.	29
AthenaEngine.Framework.Gameplay.Level	
A Level object holds all details required to handle level drawing.	30
AthenaEngine.Framework.Systems.ResourceManager< T >	
The ResourceManager class manages resources on behalf of the game.	31
AthenaEngine.Framework.Systems.LevelLoader.Xml.Sprite	32
AthenaEngine.Framework.Gameplay.Tile	
A tile is used to draw levels.	32

AthenaEngine.Framework.Systems.Triggers	
Triggers.	34
AthenaEngine.Framework.UI.UI	
UI	35
AthenaEngine.Framework.UI.UIButton	
User interface button.	36

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Athena/Athena/Athena/Debug.cs	39
Athena/Athena/Athena/Game1.cs	39
Athena/Athena/Athena/Program.cs	39
Athena/Athena/Athena/Properties/AssemblyInfo.cs	39
Athena/Athena/AthenaEngine/AthenaEngine.cs	40
Athena/Athena/AthenaEngine/Framework/Camera2D.cs	40
Athena/Athena/AthenaEngine/Framework/Gameplay/Character.cs	40
Athena/Athena/AthenaEngine/Framework/Gameplay/Level.cs	40
Athena/Athena/AthenaEngine/Framework/Gameplay/Tile.cs	41
Athena/Athena/AthenaEngine/Framework/Gameplay/RPG/Inventory.cs	41
Athena/Athena/AthenaEngine/Framework/Gameplay/RPG/Item.cs	41
Athena/Athena/AthenaEngine/Framework/Gameplay/RPG/ItemInstance.cs	41
Athena/Athena/AthenaEngine/Framework/Interfaces/ICollidable.cs	42
Athena/Athena/AthenaEngine/Framework/Interfaces/IDrawable.cs	42
Athena/Athena/AthenaEngine/Framework/Interfaces/IFocusable.cs	42
Athena/Athena/AthenaEngine/Framework/Interfaces/IMoveable.cs	42
Athena/Athena/AthenaEngine/Framework/Primitives/BoundingBox2D.cs	43
Athena/Athena/AthenaEngine/Framework/Primitives/CollidableEntity.cs	43
Athena/Athena/AthenaEngine/Framework/Primitives/Directions.cs	43
Athena/Athena/AthenaEngine/Framework/Primitives/DrawableEntity.cs	43
Athena/Athena/AthenaEngine/Framework/Primitives/Entity.cs	44
Athena/Athena/AthenaEngine/Framework/Systems/LevelLoaderXml.cs	44
Athena/Athena/AthenaEngine/Framework/Systems/ResourceManager.cs	44
Athena/Athena/AthenaEngine/Framework/Systems/Triggers.cs	44
Athena/Athena/AthenaEngine/Framework/UI/UI.cs	45
Athena/Athena/AthenaEngine/Framework/UI/UIButton.cs	45
Athena/Athena/AthenaEngine/obj/Debug/TemporaryGeneratedFile_036C0B5B-1481-4323-8D20-8F5AD-CB23D92.cs	45
Athena/Athena/AthenaEngine/obj/Debug/TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.-cs	45
Athena/Athena/AthenaEngine/obj/Debug/TemporaryGeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70-B10BC5D3.cs	45
Athena/Athena/AthenaEngine/Properties/AssemblyInfo.cs	39
Athena/Athena/AthenaTest/Engine Tests/Primitives Testing/BoundingBox2DTest.cs	45
Athena/Athena/AthenaTest/obj/Debug/TemporaryGeneratedFile_036C0B5B-1481-4323-8D20-8F5ADC-B23D92.cs	45
Athena/Athena/AthenaTest/obj/Debug/TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.-cs	45

Athena/Athena/AthenaTest/obj/Debug/ TemporaryGeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10-BC5D3.cs	45
Athena/Athena/AthenaTest/Properties/ AssemblyInfo.cs	39

Chapter 5

Namespace Documentation

5.1 Package Athena

Classes

- class **Debug**
- class [Game1](#)

This is the main type for your game

5.2 Package AthenaEngine

Namespaces

- package [Framework](#)

Classes

- class [AthenaEngine](#)

5.3 Package AthenaEngine.Framework

Namespaces

- package [Gameplay](#)
- package [Interfaces](#)
- package [Primitives](#)
- package [Systems](#)
- package [UI](#)

Classes

- class [Camera2D](#)
Camera2D

5.4 Package AthenaEngine.Framework.Gameplay

Namespaces

- package [RPG](#)

Classes

- class [Character](#)
A character class holds important detail about each character such as their items, level, experience, skills, etc.
- class [Level](#)
A [Level](#) object holds all details required to handle level drawing.
- class [Tile](#)
A tile is used to draw levels.

5.5 Package AthenaEngine.Framework.Gameplay.RPG

Classes

- class [Inventory](#)
[Inventory](#).
- class [Item](#)
A gameplay item can be held by a character.
- class [ItemInstance](#)
An item instance is a particular instance of an item.

5.6 Package AthenaEngine.Framework.Interfaces

Classes

- interface [ICollidable< T >](#)
I collidable.
- interface [IDrawable](#)
I drawable.
- interface [IFocusable](#)
I focusable.
- interface [IMoveable](#)

5.7 Package AthenaEngine.Framework.Primitives

Classes

- class [BoundingBox2D](#)
[BoundingBox2D](#) is used for bounding boxes on 2D objects.
- class [CollidableEntity](#)
The [Entity](#) class is used to store objects that have positions.
- class [Directions](#)
- class [DrawableEntity](#)

This is an entity which can be drawn.

- class [Entity](#)

The [Entity](#) class is the superclass for anything.

5.8 Package AthenaEngine.Framework.Systems

Classes

- class **LevelLoaderXml**

the LevelLoader is used to load levels.

- class [ResourceManager](#)< T >

The ResourceManager class manages resources on behalf of the game.

- class [Triggers](#)

Triggers.

5.9 Package AthenaEngine.Framework.UI

Classes

- class [UI](#)

UI

- class [UIButton](#)

User interface button.

5.10 Package AthenaTest

Namespaces

- package [Engine_Tests](#)

5.11 Package AthenaTest.Engine_Tests

Namespaces

- package [Primitives_Testing](#)

5.12 Package AthenaTest.Engine_Tests.Primitives_Testing

Classes

- class [BoundingBox2DTest](#)

Chapter 6

Class Documentation

6.1 AthenaEngine.AthenaEngine Class Reference

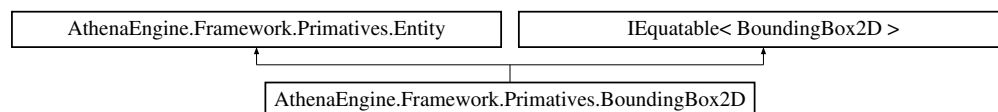
The documentation for this class was generated from the following file:

- Athena/Athena/AthenaEngine/[AthenaEngine.cs](#)

6.2 AthenaEngine.Framework.Primitives.BoundingBox2D Class Reference

[BoundingBox2D](#) is used for bounding boxes on 2D objects.

Inheritance diagram for AthenaEngine.Framework.Primitives.BoundingBox2D:



Public Member Functions

- [BoundingBox2D](#) (Vector2 [Position](#), Vector2 [Size](#))
Constructor for [BoundingBox2D](#)
- bool [Equals](#) ([BoundingBox2D](#) otherBounds)
Check if the [BoundingBox2D](#) is equal to another [BoundingBox2D](#)
- bool [CollidesWith](#) ([BoundingBox2D](#) other)
Check to see if this [BoundingBox2D](#) collides with an other [BoundingBox](#).

Properties

- [Rectangle Bounds](#) [get, set]

Additional Inherited Members

6.2.1 Detailed Description

[BoundingBox2D](#) is used for bounding boxes on 2D objects.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 AthenaEngine.Framework.Primitives.BoundingBox2D.BoundingBox2D (Vector2 *Position*, Vector2 *Size*)

Constructor for [BoundingBox2D](#)

Parameters

<i>min</i>	x/y coordinates for the bounding box
<i>max</i>	width/height for the bounding box

6.2.3 Member Function Documentation

6.2.3.1 bool AthenaEngine.Framework.Primitives.BoundingBox2D.CollidesWith (BoundingBox2D *other*)

Check to see if this [BoundingBox2D](#) collides with an other BoundingBox.

Parameters

<i>otherBounds</i>	The other BoundingBox2D to compare with.
--------------------	--

Returns

6.2.3.2 bool AthenaEngine.Framework.Primitives.BoundingBox2D.Equals (BoundingBox2D *otherBounds*)

Check if the [BoundingBox2D](#) is equal to another [BoundingBox2D](#)

Parameters

<i>otherBounds</i>	The other BoundingBox2D to check against.
--------------------	---

Returns

6.2.4 Property Documentation

6.2.4.1 Rectangle AthenaEngine.Framework.Primitives.BoundingBox2D.Bounds [get], [set], [protected]

The documentation for this class was generated from the following file:

- Athena/Athena/AthenaEngine/Framework/Primitives/[BoundingBox2D.cs](#)

6.3 AthenaTest.Engine.Tests.Primitives_Testing.BoundingBox2DTest Class Reference

Public Member Functions

- void [Equality_WithSameRectangle_IsEqual](#) ()
- void [Equality_WithIdenticalRectangle_IsEqual](#) ()
- void [Equality_WithDifferentRectangle_IsNotEqual](#) ()
- void [Equality_WithItself_IsEqual](#) ()

- void [Collisions_WithItself_IsTrue](#) ()
- void [Collisions_WithTotallyDifferentRectangle_IsNotTrue](#) ()
- void [Collisions_WithEnvelopedRectangle_IsTrue](#) ()
- void [Collisions_WithRectangleOnLeft_IsFalse](#) ()
- void [Collisions_WithRectangleOnRight_IsFalse](#) ()
- void [Collisions_WithRectangleOnTop_IsFalse](#) ()
- void [Collisions_WithRectangleOnBottom_IsFalse](#) ()
- void [Collisions_WithRectangleCollidingLeft_IsTrue](#) ()
- void [Collisions_WithRectangleCollidingRight_IsTrue](#) ()
- void [Collisions_WithRectangleCollidingTop_IsTrue](#) ()
- void [Collisions_WithRectangleCollidingBottom_IsTrue](#) ()

6.3.1 Member Function Documentation

- 6.3.1.1 void AthenaTest.Engine_Tests.Primitives_Testing.BoundingBox2DTest.Collisions_WithEnvelopedRectangle_IsTrue ()
- 6.3.1.2 void AthenaTest.Engine_Tests.Primitives_Testing.BoundingBox2DTest.Collisions_WithItself_IsTrue ()
- 6.3.1.3 void AthenaTest.Engine_Tests.Primitives_Testing.BoundingBox2DTest.Collisions_WithRectangleCollidingBottom_IsTrue ()
- 6.3.1.4 void AthenaTest.Engine_Tests.Primitives_Testing.BoundingBox2DTest.Collisions_WithRectangleCollidingLeft_IsTrue ()
- 6.3.1.5 void AthenaTest.Engine_Tests.Primitives_Testing.BoundingBox2DTest.Collisions_WithRectangleCollidingRight_IsTrue ()
- 6.3.1.6 void AthenaTest.Engine_Tests.Primitives_Testing.BoundingBox2DTest.Collisions_WithRectangleCollidingTop_IsTrue ()
- 6.3.1.7 void AthenaTest.Engine_Tests.Primitives_Testing.BoundingBox2DTest.Collisions_WithRectangleOnBottom_IsFalse ()
- 6.3.1.8 void AthenaTest.Engine_Tests.Primitives_Testing.BoundingBox2DTest.Collisions_WithRectangleOnLeft_IsFalse ()
- 6.3.1.9 void AthenaTest.Engine_Tests.Primitives_Testing.BoundingBox2DTest.Collisions_WithRectangleOnRight_IsFalse ()
- 6.3.1.10 void AthenaTest.Engine_Tests.Primitives_Testing.BoundingBox2DTest.Collisions_WithRectangleOnTop_IsFalse ()
- 6.3.1.11 void AthenaTest.Engine_Tests.Primitives_Testing.BoundingBox2DTest.Collisions_WithTotallyDifferentRectangle_IsNotTrue ()
- 6.3.1.12 void AthenaTest.Engine_Tests.Primitives_Testing.BoundingBox2DTest.Equality_WithDifferentRectangle_IsNotEqual ()
- 6.3.1.13 void AthenaTest.Engine_Tests.Primitives_Testing.BoundingBox2DTest.Equality_WithIdenticalRectangle_IsEqual ()
- 6.3.1.14 void AthenaTest.Engine_Tests.Primitives_Testing.BoundingBox2DTest.Equality_WithItself_IsEqual ()
- 6.3.1.15 void AthenaTest.Engine_Tests.Primitives_Testing.BoundingBox2DTest.Equality_WithSameRectangle_IsEqual ()

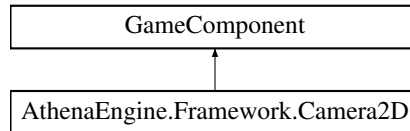
The documentation for this class was generated from the following file:

- Athena/Athena/AthenaTest/Engine Tests/Primitives Testing/[BoundingBox2DTest.cs](#)

6.4 AthenaEngine.Framework.Camera2D Class Reference

Camera2D

Inheritance diagram for AthenaEngine.Framework.Camera2D:



Public Member Functions

- [Camera2D](#) (Game game)
Initializes a new instance of the [AthenaEngine.Framework.Camera2D](#) class.
- override void [Initialize](#) ()
Called when the GameComponent needs to be initialized.
- override void [Update](#) (GameTime gameTime)
Update the specified gameTime.
- bool [IsInView](#) (Vector2 position, Texture2D texture)
Determines whether the target is in view given the specified position. This can be used to increase performance by not drawing objects directly in the viewport

Protected Attributes

- float [_viewportHeight](#)
- float [_viewportWidth](#)

Properties

- Vector2 [Position](#) [get, set]
Gets or sets the position.
- float [Rotation](#) [get, set]
- Vector2 [Origin](#) [get, set]
- float [Scale](#) [get, set]
- Vector2 [ScreenCenter](#) [get, set]
- Matrix [Transform](#) [get, set]
- IFocusable [Focus](#) [get, set]
- float [MoveSpeed](#) [get, set]

6.4.1 Detailed Description

Camera2D

6.4.2 Constructor & Destructor Documentation

6.4.2.1 AthenaEngine.Framework.Camera2D.Camera2D (Game game)

Initializes a new instance of the [AthenaEngine.Framework.Camera2D](#) class.

Parameters

<i>game</i>	Game.
-------------	-------

6.4.3 Member Function Documentation

6.4.3.1 override void AthenaEngine.Framework.Camera2D.Initialize ()

Called when the GameComponent needs to be initialized.

6.4.3.2 bool AthenaEngine.Framework.Camera2D.IsInView (Vector2 *position*, Texture2D *texture*)

Determines whether the target is in view given the specified position. This can be used to increase performance by not drawing objects directly in the viewport

Parameters

<i>position</i>	The position.
<i>texture</i>	The texture.

Returns

`true` if [is in view] [the specified position]; otherwise, `false`.

6.4.3.3 override void AthenaEngine.Framework.Camera2D.Update (GameTime *gameTime*)

Update the specified gameTime.

Parameters

<i>gameTime</i>	Game time.
-----------------	------------

6.4.4 Member Data Documentation

6.4.4.1 float AthenaEngine.Framework.Camera2D._viewportHeight [protected]

6.4.4.2 float AthenaEngine.Framework.Camera2D._viewportWidth [protected]

6.4.5 Property Documentation

6.4.5.1 IFocusable AthenaEngine.Framework.Camera2D.Focus [get], [set]

6.4.5.2 float AthenaEngine.Framework.Camera2D.MoveSpeed [get], [set]

6.4.5.3 Vector2 AthenaEngine.Framework.Camera2D.Origin [get], [set]

6.4.5.4 Vector2 AthenaEngine.Framework.Camera2D.Position [get], [set]

Gets or sets the position.

The position.

6.4.5.5 float AthenaEngine.Framework.Camera2D.Rotation [get], [set]

6.4.5.6 float `AthenaEngine.Framework.Camera2D.Scale` `[get]`, `[set]`

6.4.5.7 Vector2 `AthenaEngine.Framework.Camera2D.ScreenCenter` `[get]`, `[set]`

6.4.5.8 Matrix `AthenaEngine.Framework.Camera2D.Transform` `[get]`, `[set]`

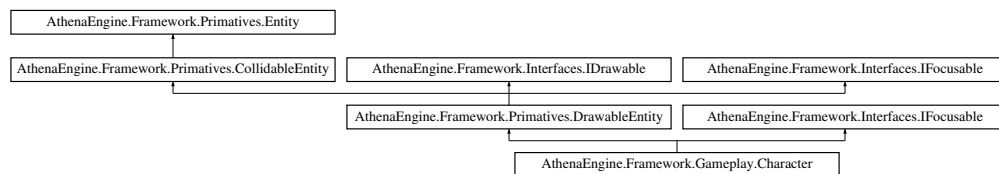
The documentation for this class was generated from the following file:

- `Athena/Athena/AthenaEngine/Framework/Camera2D.cs`

6.5 AthenaEngine.Framework.Gameplay.Character Class Reference

A character class holds important detail about each character such as their items, level, experience, skills, etc.

Inheritance diagram for `AthenaEngine.Framework.Gameplay.Character`:



Public Member Functions

- [Character](#) (Vector2 position, Vector2 size, SpriteBatch spriteBatch, Texture2D texture, [Level level](#))
Constructor for the [Character](#) class

Additional Inherited Members

6.5.1 Detailed Description

A character class holds important detail about each character such as their items, level, experience, skills, etc.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 `AthenaEngine.Framework.Gameplay.Character.Character` (Vector2 *position*, Vector2 *size*, SpriteBatch *spriteBatch*, Texture2D *texture*, Level *level*)

Constructor for the [Character](#) class

Parameters

<i>position</i>	The coordinates of the character
<i>size</i>	The size of the character
<i>spriteBatch</i>	Which spritebatch will draw the character
<i>texture</i>	What is the texture for the character

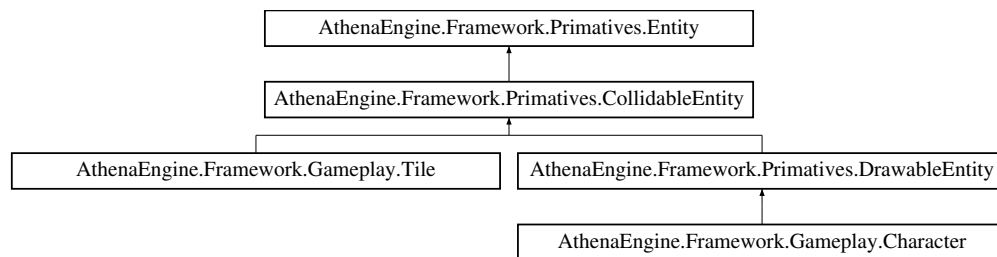
The documentation for this class was generated from the following file:

- `Athena/Athena/AthenaEngine/Framework/Gameplay/Character.cs`

6.6 AthenaEngine.Framework.Primitives.CollidableEntity Class Reference

The [Entity](#) class is used to store objects that have positions.

Inheritance diagram for AthenaEngine.Framework.Primitives.CollidableEntity:



Public Member Functions

- [CollidableEntity](#) (Vector2 position, Vector2 size, [Level](#) level)
Constructor for the [CollidableEntity](#) class.
- bool [CollidesWith](#) ([CollidableEntity](#) entity)
Check if the [CollidableEntity](#) collides with another [CollidableEntity](#).
- bool [CanMove](#) (int direction)
Determines whether this instance can move the specified direction.
- bool [Move](#) (int direction)
Move the specified direction.

Public Attributes

- [Level](#) [Level](#)

Properties

- [BoundingBox2D](#) [Bounds](#) [get, set]

Additional Inherited Members

6.6.1 Detailed Description

The [Entity](#) class is used to store objects that have positions.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 AthenaEngine.Framework.Primitives.CollidableEntity.CollidableEntity (Vector2 position, Vector2 size, Level level)

Constructor for the [CollidableEntity](#) class.

Parameters

<i>position</i>	x/y coordinates of the entity.
<i>size</i>	width/height of the entity.

6.6.3 Member Function Documentation

6.6.3.1 `bool AthenaEngine.Framework.Primitives.CollidableEntity.CanMove (int direction)`

Determines whether this instance can move the specified direction.

Returns

`true` if this instance can move the specified direction; otherwise, `false`.

Parameters

<i>direction</i>	If set to <code>true</code> direction.
------------------	--

6.6.3.2 `bool AthenaEngine.Framework.Primitives.CollidableEntity.CollidesWith (CollidableEntity entity)`

Check if the [CollidableEntity](#) collides with another [CollidableEntity](#).

Parameters

<i>entity</i>	The entity to check against
---------------	-----------------------------

Returns

Returns true if it does collide, otherwise false.

6.6.3.3 `bool AthenaEngine.Framework.Primitives.CollidableEntity.Move (int direction)`

Move the specified direction.

Parameters

<i>direction</i>	If set to <code>true</code> direction.
------------------	--

6.6.4 Member Data Documentation

6.6.4.1 `Level AthenaEngine.Framework.Primitives.CollidableEntity.Level`

6.6.5 Property Documentation

6.6.5.1 `BoundingBox2D AthenaEngine.Framework.Primitives.CollidableEntity.Bounds` `[get], [set], [protected]`

The documentation for this class was generated from the following file:

- [Athena/Athena/AthenaEngine/Framework/Primitives/CollidableEntity.cs](#)

6.7 AthenaEngine.Framework.Primitives.Directions Class Reference

Public Attributes

- `const int LEFT = 0`

- const int [UP](#) = 1
- const int [RIGHT](#) = 2
- const int [DOWN](#) = 3

6.7.1 Member Data Documentation

6.7.1.1 const int AthenaEngine.Framework.Primitives.Directions.DOWN = 3

6.7.1.2 const int AthenaEngine.Framework.Primitives.Directions.LEFT = 0

6.7.1.3 const int AthenaEngine.Framework.Primitives.Directions.RIGHT = 2

6.7.1.4 const int AthenaEngine.Framework.Primitives.Directions.UP = 1

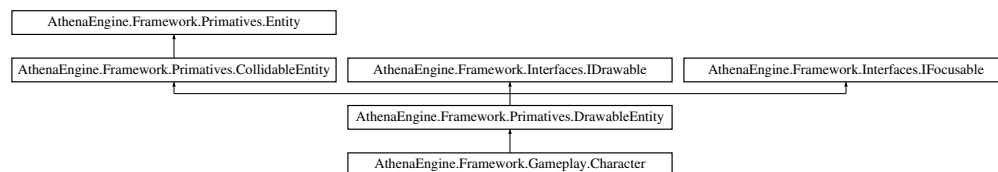
The documentation for this class was generated from the following file:

- Athena/Athena/AthenaEngine/Framework/Primitives/[Directions.cs](#)

6.8 AthenaEngine.Framework.Primitives.DrawableEntity Class Reference

This is an entity which can be drawn.

Inheritance diagram for AthenaEngine.Framework.Primitives.DrawableEntity:



Public Member Functions

- [DrawableEntity](#) (Vector2 position, Vector2 size, [Rectangle SpriteSource](#), SpriteBatch spriteBatch, Texture2D tileset, [Level level](#))

This is the constructor for the [DrawableEntity](#) class.

- void [Draw](#) ()

This draws the [DrawableEntity](#).

Public Attributes

- Color [SpriteColor](#)
- [Level level](#)

Protected Attributes

- SpriteBatch [SpriteController](#)
- Texture2D [SpriteSheet](#)
- [Rectangle SpriteSource](#)

Properties

- Vector2 [Position](#) [get, set]

Additional Inherited Members

6.8.1 Detailed Description

This is an entity which can be drawn.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 `AthenaEngine.Framework.Primitives.DrawableEntity.DrawableEntity (Vector2 position, Vector2 size, Rectangle SpriteSource, SpriteBatch spriteBatch, Texture2D tileset, Level level)`

This is the constructor for the [DrawableEntity](#) class.

Parameters

<i>position</i>	Where the DrawableEntity will start
<i>size</i>	The size in pixels of the DrawableEntity
<i>spriteBatch</i>	The SpriteBatch responsible for drawing the entity
<i>texture</i>	The texture used to draw the entity

6.8.3 Member Function Documentation

6.8.3.1 `void AthenaEngine.Framework.Primitives.DrawableEntity.Draw ()`

This draws the [DrawableEntity](#).

Implements [AthenaEngine.Framework.Interfaces.IDrawable](#).

6.8.4 Member Data Documentation

6.8.4.1 `Level AthenaEngine.Framework.Primitives.DrawableEntity.level`

6.8.4.2 `Color AthenaEngine.Framework.Primitives.DrawableEntity.SpriteColor`

6.8.4.3 `SpriteBatch AthenaEngine.Framework.Primitives.DrawableEntity.SpriteController` [protected]

6.8.4.4 `Texture2D AthenaEngine.Framework.Primitives.DrawableEntity.SpriteSheet` [protected]

6.8.4.5 `Rectangle AthenaEngine.Framework.Primitives.DrawableEntity.SpriteSource` [protected]

6.8.5 Property Documentation

6.8.5.1 `Vector2 AthenaEngine.Framework.Primitives.DrawableEntity.Position` [get], [set]

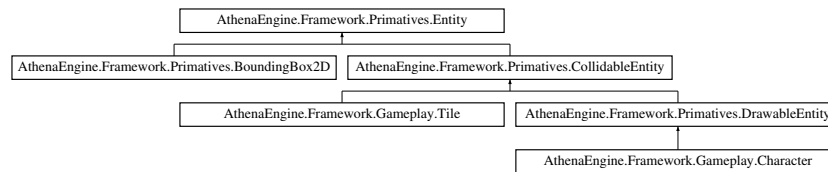
The documentation for this class was generated from the following file:

- `Athena/Athena/AthenaEngine/Framework/Primitives/DrawableEntity.cs`

6.9 AthenaEngine.Framework.Primitives.Entity Class Reference

The [Entity](#) class is the superclass for anything.

Inheritance diagram for AthenaEngine.Framework.Primitives.Entity:



Protected Member Functions

- void [Move](#) (int direction)

Protected Attributes

- Vector2 [Position](#)
- Vector2 [Size](#)

Properties

- Rectangle [Rectangle](#) [get, set]
Gets or sets the rectangle.
- int [X](#) [get, set]
- int [Y](#) [get, set]
- int [Width](#) [get, set]
- int [Height](#) [get, set]

6.9.1 Detailed Description

The [Entity](#) class is the superclass for anything.

6.9.2 Member Function Documentation

6.9.2.1 void AthenaEngine.Framework.Primitives.Entity.Move (int *direction*) [protected]

6.9.3 Member Data Documentation

6.9.3.1 Vector2 AthenaEngine.Framework.Primitives.Entity.Position [protected]

6.9.3.2 Vector2 AthenaEngine.Framework.Primitives.Entity.Size [protected]

6.9.4 Property Documentation

6.9.4.1 int AthenaEngine.Framework.Primitives.Entity.Height [get], [set]

6.9.4.2 Rectangle AthenaEngine.Framework.Primitives.Entity.Rectangle [get], [set], [protected]

Gets or sets the rectangle.

The rectangle.

6.9.4.3 `int AthenaEngine.Framework.Primitives.Entity.Width` `[get]`, `[set]`

6.9.4.4 `int AthenaEngine.Framework.Primitives.Entity.X` `[get]`, `[set]`

6.9.4.5 `int AthenaEngine.Framework.Primitives.Entity.Y` `[get]`, `[set]`

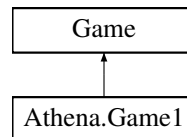
The documentation for this class was generated from the following file:

- [Athena/Athena/AthenaEngine/Framework/Primitives/Entity.cs](#)

6.10 Athena.Game1 Class Reference

This is the main type for your game

Inheritance diagram for `Athena.Game1`:



Public Member Functions

- [Game1](#) ()
The actual game class constructor.

Public Attributes

- SpriteFont [font](#)

Protected Member Functions

- override void [Initialize](#) ()
Allows the game to perform any initialization it needs to before starting to run. This is where it can query for any required services and load any non-graphic related content. Calling base.Initialize will enumerate through any components and initialize them as well.
- override void [LoadContent](#) ()
LoadContent will be called once per game and is the place to load all of your content.
- override void [UnloadContent](#) ()
UnloadContent will be called once per game and is the place to unload all content.
- override void [Update](#) (GameTime gameTime)
Allows the game to run logic such as updating the world, checking for collisions, gathering input, and playing audio.
- override void [Draw](#) (GameTime gameTime)
This is called when the game should draw itself.

6.10.1 Detailed Description

This is the main type for your game

6.10.2 Constructor & Destructor Documentation

6.10.2.1 Athena.Game1.Game1 ()

The actual game class constructor.

6.10.3 Member Function Documentation

6.10.3.1 override void Athena.Game1.Draw (gameTime *gameTime*) [protected]

This is called when the game should draw itself.

Parameters

<i>gameTime</i>	Provides a snapshot of timing values.
-----------------	---------------------------------------

6.10.3.2 override void Athena.Game1.Initialize () [protected]

Allows the game to perform any initialization it needs to before starting to run. This is where it can query for any required services and load any non-graphic related content. Calling base.Initialize will enumerate through any components and initialize them as well.

6.10.3.3 override void Athena.Game1.LoadContent () [protected]

LoadContent will be called once per game and is the place to load all of your content.

6.10.3.4 override void Athena.Game1.UnloadContent () [protected]

UnloadContent will be called once per game and is the place to unload all content.

6.10.3.5 override void Athena.Game1.Update (gameTime *gameTime*) [protected]

Allows the game to run logic such as updating the world, checking for collisions, gathering input, and playing audio.

Parameters

<i>gameTime</i>	Provides a snapshot of timing values.
-----------------	---------------------------------------

6.10.4 Member Data Documentation

6.10.4.1 SpriteFont Athena.Game1.font

The documentation for this class was generated from the following file:

- Athena/Athena/Athena/[Game1.cs](#)

6.11 AthenaEngine.Framework.Interfaces.ICollidable< T > Interface Template Reference

I collidable.

Public Member Functions

- bool [CollidesWith](#) (T type)

6.11.1 Detailed Description

I collidable.

6.11.2 Member Function Documentation

6.11.2.1 bool AthenaEngine.Framework.Interfaces.ICollidable< T >.CollidesWith (T type)

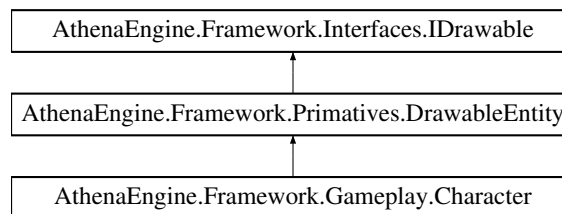
The documentation for this interface was generated from the following file:

- Athena/Athena/AthenaEngine/Framework/Interfaces/[ICollidable.cs](#)

6.12 AthenaEngine.Framework.Interfaces.IDrawable Interface Reference

I drawable.

Inheritance diagram for AthenaEngine.Framework.Interfaces.IDrawable:



Public Member Functions

- void [Draw](#) ()
Draw this instance.

6.12.1 Detailed Description

I drawable.

6.12.2 Member Function Documentation

6.12.2.1 void AthenaEngine.Framework.Interfaces.IDrawable.Draw ()

Draw this instance.

Implemented in [AthenaEngine.Framework.Primitives.DrawableEntity](#).

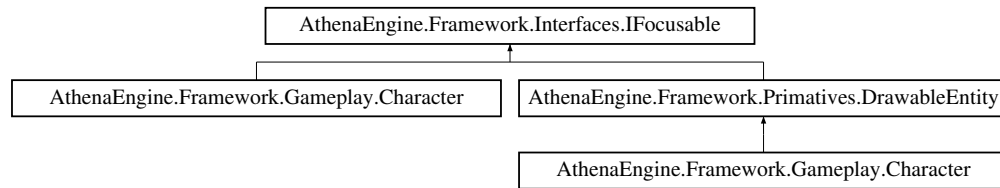
The documentation for this interface was generated from the following file:

- Athena/Athena/AthenaEngine/Framework/Interfaces/[IDrawable.cs](#)

6.13 AthenaEngine.Framework.Interfaces.IFocusable Interface Reference

I focusable.

Inheritance diagram for AthenaEngine.Framework.Interfaces.IFocusable:



Properties

- Vector2 [Position](#) [get]
Gets the position.

6.13.1 Detailed Description

I focusable.

6.13.2 Property Documentation

6.13.2.1 Vector2 AthenaEngine.Framework.Interfaces.IFocusable.Position [get]

Gets the position.

The position.

The documentation for this interface was generated from the following file:

- [Athena/Athena/AthenaEngine/Framework/Interfaces/IFocusable.cs](#)

6.14 AthenaEngine.Framework.Interfaces.IMoveable Interface Reference

Public Member Functions

- bool [Move](#) (string direction)
- bool [CanMove](#) (string direction)

6.14.1 Member Function Documentation

6.14.1.1 bool AthenaEngine.Framework.Interfaces.IMoveable.CanMove (string direction)

6.14.1.2 bool AthenaEngine.Framework.Interfaces.IMoveable.Move (string direction)

The documentation for this interface was generated from the following file:

- [Athena/Athena/AthenaEngine/Framework/Interfaces/IMoveable.cs](#)

6.15 AthenaEngine.Framework.Gameplay.RPG.Inventory Class Reference

[Inventory.](#)

Public Member Functions

- void [Add](#) ([Item](#) item, int quantity)
Add the specified item and quantity.

6.15.1 Detailed Description

[Inventory.](#)

6.15.2 Member Function Documentation

6.15.2.1 void AthenaEngine.Framework.Gameplay.RPG.Inventory.Add ([Item](#) item, int quantity)

Add the specified item and quantity.

Parameters

<i>item</i>	Item .
<i>quantity</i>	Quantity.

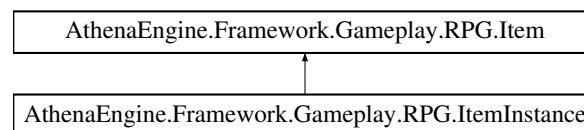
The documentation for this class was generated from the following file:

- Athena/Athena/AthenaEngine/Framework/Gameplay/RPG/[Inventory.cs](#)

6.16 AthenaEngine.Framework.Gameplay.RPG.Item Class Reference

A gameplay item can be held by a character.

Inheritance diagram for AthenaEngine.Framework.Gameplay.RPG.Item:



Public Member Functions

- [Item](#) (string name)
Constructor for the item class

Properties

- string [Name](#) [get, set]
The name of the item

6.16.1 Detailed Description

A gameplay item can be held by a character.

6.16.2 Constructor & Destructor Documentation

6.16.2.1 AthenaEngine.Framework.Gameplay.RPG.Item.Item (string *name*)

Constructor for the item class

Parameters

<i>name</i>	Name of the item
-------------	------------------

6.16.3 Property Documentation

6.16.3.1 string AthenaEngine.Framework.Gameplay.RPG.Item.Name [get], [set]

The name of the item

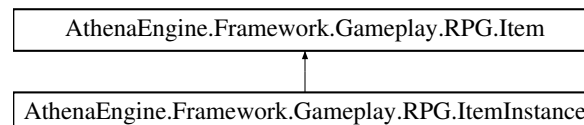
The documentation for this class was generated from the following file:

- Athena/Athena/AthenaEngine/Framework/Gameplay/RPG/[Item.cs](#)

6.17 AthenaEngine.Framework.Gameplay.RPG.ItemInstance Class Reference

An item instance is a particular instance of an item.

Inheritance diagram for AthenaEngine.Framework.Gameplay.RPG.ItemInstance:



Public Member Functions

- [ItemInstance](#) ([Item](#) item, int quantity)
Constructor for the [ItemInstance](#) class.

Additional Inherited Members

6.17.1 Detailed Description

An item instance is a particular instance of an item.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 AthenaEngine.Framework.Gameplay.RPG.ItemInstance.ItemInstance ([Item](#) *item*, int *quantity*)

Constructor for the [ItemInstance](#) class.

Parameters

<i>item</i>	The actual item of which this is an instance
<i>quantity</i>	How many of that item are in this particular instance?

The documentation for this class was generated from the following file:

- [Athena/Athena/AthenaEngine/Framework/Gameplay/RPG/ItemInstance.cs](#)

6.18 AthenaEngine.Framework.Gameplay.Level Class Reference

A [Level](#) object holds all details required to handle level drawing.

Public Member Functions

- [Level](#) (string levelName, SpriteBatch spriteBatch, ResourceManager< Texture2D > resourceManager)
Object constructor for the [Level](#) class.
- void [Draw](#) ()
Draw the level.

Public Attributes

- List< [Tile](#) > [TileList](#)

6.18.1 Detailed Description

A [Level](#) object holds all details required to handle level drawing.

6.18.2 Constructor & Destructor Documentation

6.18.2.1 [AthenaEngine.Framework.Gameplay.Level.Level](#) (string *levelName*, SpriteBatch *spriteBatch*, ResourceManager< Texture2D > *resourceManager*)

Object constructor for the [Level](#) class.

Parameters

<i>levelName</i>	The name of the level to load.
<i>spriteBatch</i>	The spriteBatch to draw the level with.
<i>resource-Manager</i>	The resourceManager handling the games' textures.

6.18.3 Member Function Documentation

6.18.3.1 void [AthenaEngine.Framework.Gameplay.Level.Draw](#) ()

Draw the level.

6.18.4 Member Data Documentation

6.18.4.1 List<Tile> AthenaEngine.Framework.Gameplay.Level.TileList

The documentation for this class was generated from the following file:

- Athena/Athena/AthenaEngine/Framework/Gameplay/Level.cs

6.19 AthenaEngine.Framework.Systems.ResourceManager< T > Class Template Reference

The ResourceManager class manages resources on behalf of the game.

Public Member Functions

- [ResourceManager](#) (Game game)
Creates a new ResourceManager to manage resources for a game.
- T [Add](#) (string key, T resource)
Add a new resource to the resources list maintained by the ResourceManager.
- T [Get](#) (string key)
Gets the resource stored associated with a key.

6.19.1 Detailed Description

The ResourceManager class manages resources on behalf of the game.

Template Parameters

<i>T</i>	The type of resource to manage
----------	--------------------------------

6.19.2 Constructor & Destructor Documentation

6.19.2.1 AthenaEngine.Framework.Systems.ResourceManager< T >.ResourceManager (Game game)

Creates a new ResourceManager to manage resources for a game.

Parameters

<i>game</i>	The game to manage resources for.
-------------	-----------------------------------

6.19.3 Member Function Documentation

6.19.3.1 T AthenaEngine.Framework.Systems.ResourceManager< T >.Add (string key, T resource)

Add a new resource to the resources list maintained by the ResourceManager.

Parameters

<i>key</i>	The key to associate the resource with
<i>resource</i>	The resource associated with the key

Returns

6.19.3.2 T AthenaEngine.Framework.Systems.ResourceManager< T >.Get (string key)

Gets the resource stored associated with a key.

Parameters

<i>key</i>	The key to find the resource associated with.
------------	---

Returns

The documentation for this class was generated from the following file:

- Athena/Athena/AthenaEngine/Framework/Systems/[ResourceManager.cs](#)

6.20 AthenaEngine.Framework.Systems.LevelLoaderXml.Sprite Class Reference

Public Member Functions

- [Sprite](#) (int x, int y, string name, bool collides)

Public Attributes

- int [X](#)
- int [Y](#)
- string [Name](#)
- bool [Collides](#)

6.20.1 Constructor & Destructor Documentation

6.20.1.1 AthenaEngine.Framework.Systems.LevelLoaderXml.Sprite.Sprite (int x, int y, string name, bool collides)

6.20.2 Member Data Documentation

6.20.2.1 bool AthenaEngine.Framework.Systems.LevelLoaderXml.Sprite.Collides

6.20.2.2 string AthenaEngine.Framework.Systems.LevelLoaderXml.Sprite.Name

6.20.2.3 int AthenaEngine.Framework.Systems.LevelLoaderXml.Sprite.X

6.20.2.4 int AthenaEngine.Framework.Systems.LevelLoaderXml.Sprite.Y

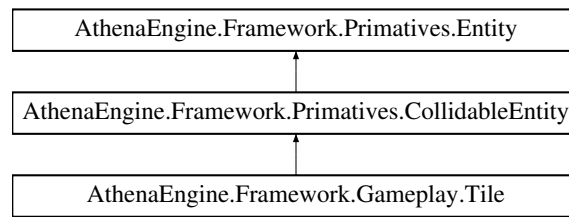
The documentation for this class was generated from the following file:

- Athena/Athena/AthenaEngine/Framework/Systems/[LevelLoaderXml.cs](#)

6.21 AthenaEngine.Framework.Gameplay.Tile Class Reference

A tile is used to draw levels.

Inheritance diagram for AthenaEngine.Framework.Gameplay.Tile:



Public Member Functions

- **Tile** (int x, int y, string TileSet, int xOffset, int yOffset, [Level](#) level)
Class constructor for a new [Tile](#) object.
- void **MakeDrawable** (SpriteBatch spriteBatch, ResourceManager< Texture2D > resourceManager)
Make a tile drawable by giving it a texture and a spriteBatch.
- void **Draw** ()
Draw the tile.
- void **FireTrigger** (object[] args)
Fires the trigger.
- void **AddTrigger** (string triggerName)
Adds the trigger.

Public Attributes

- bool **Collides** = false
- [DrawableEntity](#) **Sprite**
- bool **HasTrigger** = false

Additional Inherited Members

6.21.1 Detailed Description

A tile is used to draw levels.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 AthenaEngine.Framework.Gameplay.Tile.Tile (int x, int y, string *TileSet*, int *xOffset*, int *yOffset*, [Level](#) *level*)

Class constructor for a new [Tile](#) object.

Parameters

<i>x</i>	The X coordinate of the new tile.
<i>y</i>	The Y coordinate of the new tile.

6.21.3 Member Function Documentation

6.21.3.1 void AthenaEngine.Framework.Gameplay.Tile.AddTrigger (string *triggerName*)

Adds the trigger.

Parameters

<i>triggerName</i>	Trigger name.
--------------------	---------------

6.21.3.2 void AthenaEngine.Framework.Gameplay.Tile.Draw ()

Draw the tile.

6.21.3.3 void AthenaEngine.Framework.Gameplay.Tile.FireTrigger (object[] args)

Fires the trigger.

Parameters

<i>args</i>	Arguments.
-------------	------------

6.21.3.4 void AthenaEngine.Framework.Gameplay.Tile.MakeDrawable (SpriteBatch *spriteBatch*, ResourceManager< Texture2D > *resourceManager*)

Make a tile drawable by giving it a texture and a spriteBatch.

Parameters

<i>spriteBatch</i>	The SpriteBatch to draw the tile with.
<i>texture</i>	The texture to draw the tile with.

6.21.4 Member Data Documentation

6.21.4.1 bool AthenaEngine.Framework.Gameplay.Tile.Collides = false

6.21.4.2 bool AthenaEngine.Framework.Gameplay.Tile.HasTrigger = false

6.21.4.3 DrawableEntity AthenaEngine.Framework.Gameplay.Tile.Sprite

The documentation for this class was generated from the following file:

- Athena/Athena/AthenaEngine/Framework/Gameplay/[Tile.cs](#)

6.22 AthenaEngine.Framework.Systems.Triggers Class Reference

[Triggers.](#)

Static Public Member Functions

- static void [test](#) ()

This was used to test

- static void [encounter](#) ()

Random encounter test

6.22.1 Detailed Description

[Triggers](#).

6.22.2 Member Function Documentation

6.22.2.1 `static void AthenaEngine.Framework.Systems.Triggers.encounter () [static]`

Random encounter test

6.22.2.2 `static void AthenaEngine.Framework.Systems.Triggers.test () [static]`

This was used to test

The documentation for this class was generated from the following file:

- [Athena/Athena/AthenaEngine/Framework/Systems/Triggers.cs](#)

6.23 AthenaEngine.Framework.UI.UI Class Reference

[UI](#)

Public Member Functions

- [UI](#) (SpriteBatch spriteBatch, ResourceManager< Texture2D > textureManager, ResourceManager< SpriteFont > fontManager)
Initializes a new instance of the [AthenaEngine.Framework.UI.UI](#) class.
- void [AddButton](#) (Vector2 position, string label)
Adds the button.
- void [Draw](#) ()
Draw this instance.

Public Attributes

- [Level Level](#)

6.23.1 Detailed Description

[UI](#)

6.23.2 Constructor & Destructor Documentation

6.23.2.1 `AthenaEngine.Framework.UI.UI (SpriteBatch spriteBatch, ResourceManager< Texture2D > textureManager, ResourceManager< SpriteFont > fontManager)`

Initializes a new instance of the [AthenaEngine.Framework.UI.UI](#) class.

Parameters

<i>spriteBatch</i>	Sprite batch.
<i>textureManager</i>	Texture manager.
<i>fontManager</i>	Font manager.

6.23.3 Member Function Documentation

6.23.3.1 void AthenaEngine.Framework.UI.UI.AddButton (Vector2 *position*, string *label*)

Adds the button.

Parameters

<i>position</i>	Position.
<i>label</i>	Label.

6.23.3.2 void AthenaEngine.Framework.UI.UI.Draw ()

Draw this instance.

6.23.4 Member Data Documentation

6.23.4.1 Level AthenaEngine.Framework.UI.UI.Level

The documentation for this class was generated from the following file:

- [Athena/Athena/AthenaEngine/Framework/UI/UI.cs](#)

6.24 AthenaEngine.Framework.UI.UIButton Class Reference

User interface button.

Public Member Functions

- [UIButton](#) (Rectangle *rectangle*, SpriteBatch *spriteBatch*, Texture2D *texture*, Color *color*, string *label*, SpriteFont *font*)
Initializes a new instance of the [AthenaEngine.Framework.UI.UIButton](#) class.
- void [Draw](#) ()
Draw this instance.

Public Attributes

- [Level](#) [Level](#)

6.24.1 Detailed Description

User interface button.

6.24.2 Constructor & Destructor Documentation

6.24.2.1 AthenaEngine.Framework.UI.UIButton.UIButton (Rectangle *rectangle*, SpriteBatch *spriteBatch*, Texture2D *texture*, Color *color*, string *label*, SpriteFont *font*)

Initializes a new instance of the [AthenaEngine.Framework.UI.UIButton](#) class.

Parameters

<i>rectangle</i>	Rectangle.
<i>spriteBatch</i>	Sprite batch.
<i>texture</i>	Texture.
<i>color</i>	Color.
<i>label</i>	Label.
<i>font</i>	Font.

6.24.3 Member Function Documentation

6.24.3.1 void AthenaEngine.Framework.UI.UIButton.Draw ()

Draw this instance.

6.24.4 Member Data Documentation

6.24.4.1 Level AthenaEngine.Framework.UI.UIButton.Level

The documentation for this class was generated from the following file:

- Athena/Athena/AthenaEngine/Framework/UI/[UIButton.cs](#)

Chapter 7

File Documentation

7.1 Athena/Athena/Athena/Debug.cs File Reference

Classes

- class **Athena.Debug**

Namespaces

- package [Athena](#)

7.2 Athena/Athena/Athena/Game1.cs File Reference

Classes

- class [Athena.Game1](#)
This is the main type for your game

Namespaces

- package [Athena](#)

7.3 Athena/Athena/Athena/Program.cs File Reference

Namespaces

- package [Athena](#)

7.4 Athena/Athena/Athena/Properties/AssemblyInfo.cs File Reference

7.5 Athena/Athena/AthenaEngine/Properties/AssemblyInfo.cs File Reference

7.6 Athena/Athena/AthenaTest/Properties/AssemblyInfo.cs File Reference

7.7 Athena/Athena/AthenaEngine/AthenaEngine.cs File Reference

Classes

- class [AthenaEngine.AthenaEngine](#)

Namespaces

- package [AthenaEngine](#)

7.8 Athena/Athena/AthenaEngine/Framework/Camera2D.cs File Reference

Classes

- class [AthenaEngine.Framework.Camera2D](#)
[Camera2D](#)

Namespaces

- package [AthenaEngine.Framework](#)

7.9 Athena/Athena/AthenaEngine/Framework/Gameplay/Character.cs File Reference

Classes

- class [AthenaEngine.Framework.Gameplay.Character](#)
A character class holds important detail about each character such as their items, level, experience, skills, etc.

Namespaces

- package [AthenaEngine.Framework.Gameplay](#)

7.10 Athena/Athena/AthenaEngine/Framework/Gameplay/Level.cs File Reference

Classes

- class [AthenaEngine.Framework.Gameplay.Level](#)
A [Level](#) object holds all details required to handle level drawing.

Namespaces

- package [AthenaEngine.Framework.Gameplay](#)

7.11 Athena/Athena/AthenaEngine/Framework/Gameplay/RPG/Inventory.cs File Reference

Classes

- class [AthenaEngine.Framework.Gameplay.RPG.Inventory](#)
Inventory.

Namespaces

- package [AthenaEngine.Framework.Gameplay.RPG](#)

7.12 Athena/Athena/AthenaEngine/Framework/Gameplay/RPG/Item.cs File Reference

Classes

- class [AthenaEngine.Framework.Gameplay.RPG.Item](#)
A gameplay item can be held by a character.

Namespaces

- package [AthenaEngine.Framework.Gameplay.RPG](#)

7.13 Athena/Athena/AthenaEngine/Framework/Gameplay/RPG/ItemInstance.cs File Reference

Classes

- class [AthenaEngine.Framework.Gameplay.RPG.ItemInstance](#)
An item instance is a particular instance of an item.

Namespaces

- package [AthenaEngine.Framework.Gameplay.RPG](#)

7.14 Athena/Athena/AthenaEngine/Framework/Gameplay/Tile.cs File Reference

Classes

- class [AthenaEngine.Framework.Gameplay.Tile](#)
A tile is used to draw levels.

Namespaces

- package [AthenaEngine.Framework.Gameplay](#)

7.15 Athena/Athena/AthenaEngine/Framework/Interfaces/ICollidable.cs File Reference

Classes

- interface [AthenaEngine.Framework.Interfaces.ICollidable< T >](#)
I collidable.

Namespaces

- package [AthenaEngine.Framework.Interfaces](#)

7.16 Athena/Athena/AthenaEngine/Framework/Interfaces/IDrawable.cs File Reference

Classes

- interface [AthenaEngine.Framework.Interfaces.IDrawable](#)
I drawable.

Namespaces

- package [AthenaEngine.Framework.Interfaces](#)

7.17 Athena/Athena/AthenaEngine/Framework/Interfaces/IFocusable.cs File Reference

Classes

- interface [AthenaEngine.Framework.Interfaces.IFocusable](#)
I focusable.

Namespaces

- package [AthenaEngine.Framework.Interfaces](#)

7.18 Athena/Athena/AthenaEngine/Framework/Interfaces/IMoveable.cs File Reference

Classes

- interface [AthenaEngine.Framework.Interfaces.IMoveable](#)

Namespaces

- package [AthenaEngine.Framework.Interfaces](#)

7.19 Athena/Athena/AthenaEngine/Framework/Primitives/BoundingBox2D.cs File Reference

Classes

- class [AthenaEngine.Framework.Primitives.BoundingBox2D](#)
BoundingBox2D is used for bounding boxes on 2D objects.

Namespaces

- package [AthenaEngine.Framework.Primitives](#)

7.20 Athena/Athena/AthenaEngine/Framework/Primitives/CollidableEntity.cs File Reference

Classes

- class [AthenaEngine.Framework.Primitives.CollidableEntity](#)
The [Entity](#) class is used to store objects that have positions.

Namespaces

- package [AthenaEngine.Framework.Primitives](#)

7.21 Athena/Athena/AthenaEngine/Framework/Primitives/Directions.cs File Reference

Classes

- class [AthenaEngine.Framework.Primitives.Directions](#)

Namespaces

- package [AthenaEngine.Framework.Primitives](#)

7.22 Athena/Athena/AthenaEngine/Framework/Primitives/DrawableEntity.cs File Reference

Classes

- class [AthenaEngine.Framework.Primitives.DrawableEntity](#)
This is an entity which can be drawn.

Namespaces

- package [AthenaEngine.Framework.Primitives](#)

7.23 Athena/Athena/AthenaEngine/Framework/Primitives/Entity.cs File Reference

Classes

- class [AthenaEngine.Framework.Primitives.Entity](#)
The [Entity](#) class is the superclass for anything.

Namespaces

- package [AthenaEngine.Framework.Primitives](#)

7.24 Athena/Athena/AthenaEngine/Framework/Systems/LevelLoaderXml.cs File Reference

Classes

- class **AthenaEngine.Framework.Systems.LevelLoaderXml**
the [LevelLoader](#) is used to load levels.
- class [AthenaEngine.Framework.Systems.LevelLoaderXml.Sprite](#)

Namespaces

- package [AthenaEngine.Framework.Systems](#)

7.25 Athena/Athena/AthenaEngine/Framework/Systems/ResourceManager.cs File Reference

Classes

- class [AthenaEngine.Framework.Systems.ResourceManager< T >](#)
The [ResourceManager](#) class manages resources on behalf of the game.

Namespaces

- package [AthenaEngine.Framework.Systems](#)

7.26 Athena/Athena/AthenaEngine/Framework/Systems/Triggers.cs File Reference

Classes

- class [AthenaEngine.Framework.Systems.Triggers](#)
[Triggers](#).

Namespaces

- package [AthenaEngine.Framework.Systems](#)

7.27 Athena/Athena/AthenaEngine/Framework/UI/UI.cs File Reference

Classes

- class [AthenaEngine.Framework.UI.UI](#)
UI

Namespaces

- package [AthenaEngine.Framework.UI](#)

7.28 Athena/Athena/AthenaEngine/Framework/UI/UIButton.cs File Reference

Classes

- class [AthenaEngine.Framework.UI.UIButton](#)
User interface button.

Namespaces

- package [AthenaEngine.Framework.UI](#)

7.29 Athena/Athena/AthenaEngine/obj/Debug/TemporaryGeneratedFile_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs File Reference

7.30 Athena/Athena/AthenaTest/obj/Debug/TemporaryGeneratedFile_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs File Reference

7.31 Athena/Athena/AthenaEngine/obj/Debug/TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs File Reference

7.32 Athena/Athena/AthenaTest/obj/Debug/TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs File Reference

7.33 Athena/Athena/AthenaEngine/obj/Debug/TemporaryGeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs File Reference

7.34 Athena/Athena/AthenaTest/obj/Debug/TemporaryGeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs File Reference

7.35 Athena/Athena/AthenaTest/Engine Tests/Primitives Testing/BoundingBox2DTest.cs File Reference

Classes

- class [AthenaTest.Engine_Tests.Primitives_Testing.BoundingBox2DTest](#)

Namespaces

- package [AthenaTest.Engine_Tests.Primitives_Testing](#)

Index

- [_viewportHeight](#)
 - [AthenaEngine::Framework::Camera2D](#), [17](#)
 - [_viewportWidth](#)
 - [AthenaEngine::Framework::Camera2D](#), [17](#)
- [Add](#)
 - [AthenaEngine::Framework::Gameplay::RPG::Inventory](#), [28](#)
 - [AthenaEngine::Framework::Systems::ResourceManager< T >](#), [31](#)
- [AddButton](#)
 - [AthenaEngine::Framework::UI::UI](#), [36](#)
- [AddTrigger](#)
 - [AthenaEngine::Framework::Gameplay::Tile](#), [33](#)
- [Athena](#), [9](#)
- [Athena.Game1](#), [24](#)
- [Athena/Athena/Athena/Debug.cs](#), [39](#)
- [Athena/Athena/Athena/Game1.cs](#), [39](#)
- [Athena/Athena/Athena/Program.cs](#), [39](#)
- [Athena/Athena/Athena/Properties/AssemblyInfo.cs](#), [39](#)
- [Athena/Athena/AthenaEngine/AthenaEngine.cs](#), [40](#)
- [Athena/Athena/AthenaEngine/Framework/Camera2D.cs](#), [40](#)
- [Athena/Athena/AthenaEngine/Framework/Gameplay/Character.cs](#), [40](#)
- [Athena/Athena/AthenaEngine/Framework/Gameplay/Level.cs](#), [40](#)
- [Athena/Athena/AthenaEngine/Framework/Gameplay/RPG/Inventory.cs](#), [41](#)
- [Athena/Athena/AthenaEngine/Framework/Gameplay/RPG/Item.cs](#), [41](#)
- [Athena/Athena/AthenaEngine/Framework/Gameplay/RPG/ItemInstance.cs](#), [41](#)
- [Athena/Athena/AthenaEngine/Framework/Gameplay/Tile.cs](#), [41](#)
- [Athena/Athena/AthenaEngine/Framework/Interfaces/ICollidable.cs](#), [42](#)
- [Athena/Athena/AthenaEngine/Framework/Interfaces/IDrawable.cs](#), [42](#)
- [Athena/Athena/AthenaEngine/Framework/Interfaces/IFocusable.cs](#), [42](#)
- [Athena/Athena/AthenaEngine/Framework/Interfaces/IMoveable.cs](#), [42](#)
- [Athena/Athena/AthenaEngine/Framework/Primitives/BoundingBox2D.cs](#), [43](#)
- [Athena/Athena/AthenaEngine/Framework/Primitives/CollidableEntity.cs](#), [43](#)
- [Athena/Athena/AthenaEngine/Framework/Primitives/Directions.cs](#), [43](#)
- [Athena/Athena/AthenaEngine/Framework/Primitives/DrawableEntity.cs](#), [43](#)
- [Athena/Athena/AthenaEngine/Framework/Primitives/Entity.cs](#), [44](#)
- [Athena/Athena/AthenaEngine/Framework/Systems/LevelLoaderXml.cs](#), [44](#)
- [Athena/Athena/AthenaEngine/Framework/Systems/ResourceManager.cs](#), [44](#)
- [Athena/Athena/AthenaEngine/Framework/Systems/Triggers.cs](#), [44](#)
- [Athena/Athena/AthenaEngine/Framework/UI/UI.cs](#), [45](#)
- [Athena/Athena/AthenaEngine/Framework/UI/UIButton.cs](#), [45](#)
- [Athena/Athena/AthenaEngine/Properties/AssemblyInfo.cs](#), [39](#)
- [Athena/Athena/AthenaEngine/obj/Debug/TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs](#), [45](#)
- [Athena/Athena/AthenaTest/Engine Tests/Primitives Testing/BoundingBox2DTest.cs](#), [45](#)
- [Athena/Athena/AthenaTest/Properties/AssemblyInfo.cs](#), [39](#)
- [Athena/Athena/AthenaTest/obj/Debug/TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs](#), [45](#)
- [Athena::Game1](#)
 - [Draw](#), [25](#)
 - [font](#), [25](#)
 - [Game1](#), [25](#)
 - [Initialize](#), [25](#)
 - [LoadContent](#), [25](#)
 - [UnloadContent](#), [25](#)
 - [Update](#), [25](#)
- [AthenaEngine](#), [9](#)
- [AthenaEngine.AthenaEngine](#), [13](#)
- [AthenaEngine.Framework](#), [9](#)
- [AthenaEngine.Framework.Camera2D](#), [16](#)
- [AthenaEngine.Framework.Gameplay](#), [10](#)
- [AthenaEngine.Framework.Gameplay.Character](#), [18](#)
- [AthenaEngine.Framework.Gameplay.Level](#), [30](#)
- [AthenaEngine.Framework.Gameplay.RPG](#), [10](#)
- [AthenaEngine.Framework.Gameplay.RPG.Inventory](#), [28](#)
- [AthenaEngine.Framework.Gameplay.RPG.Item](#), [28](#)
- [AthenaEngine.Framework.Gameplay.RPG.ItemInstance](#), [29](#)
- [AthenaEngine.Framework.Gameplay.Tile](#), [32](#)
- [AthenaEngine.Framework.Interfaces](#), [10](#)
- [AthenaEngine.Framework.Interfaces.ICollidable< T >](#), [25](#)

- AthenaEngine.Framework.Interfaces.IDrawable, [26](#)
- AthenaEngine.Framework.Interfaces.IFocusable, [27](#)
- AthenaEngine.Framework.Interfaces.IMoveable, [27](#)
- AthenaEngine.Framework.Primitives, [10](#)
- AthenaEngine.Framework.Primitives.BoundingBox2D, [13](#)
- AthenaEngine.Framework.Primitives.CollidableEntity, [19](#)
- AthenaEngine.Framework.Primitives.Directions, [20](#)
- AthenaEngine.Framework.Primitives.DrawableEntity, [21](#)
- AthenaEngine.Framework.Primitives.Entity, [23](#)
- AthenaEngine.Framework.Systems, [11](#)
- AthenaEngine.Framework.Systems.LevelLoaderXml -
Sprite, [32](#)
- AthenaEngine.Framework.Systems.ResourceManager<
T >, [31](#)
- AthenaEngine.Framework.Systems.Triggers, [34](#)
- AthenaEngine.Framework.UI, [11](#)
- AthenaEngine.Framework.UI.UI, [35](#)
- AthenaEngine.Framework.UI.UIButton, [36](#)
- AthenaEngine::Framework::Camera2D
 - _viewportHeight, [17](#)
 - _viewportWidth, [17](#)
 - Camera2D, [16](#)
 - Focus, [17](#)
 - Initialize, [17](#)
 - IsInView, [17](#)
 - MoveSpeed, [17](#)
 - Origin, [17](#)
 - Position, [17](#)
 - Rotation, [17](#)
 - Scale, [17](#)
 - ScreenCenter, [18](#)
 - Transform, [18](#)
 - Update, [17](#)
- AthenaEngine::Framework::Gameplay::Character
 - Character, [18](#)
- AthenaEngine::Framework::Gameplay::Level
 - Draw, [30](#)
 - Level, [30](#)
 - TileList, [30](#)
- AthenaEngine::Framework::Gameplay::RPG::Inventory
 - Add, [28](#)
- AthenaEngine::Framework::Gameplay::RPG::Item
 - Item, [29](#)
 - Name, [29](#)
- AthenaEngine::Framework::Gameplay::RPG::Item-
Instance
 - ItemInstance, [29](#)
- AthenaEngine::Framework::Gameplay::Tile
 - AddTrigger, [33](#)
 - Collides, [34](#)
 - Draw, [34](#)
 - FireTrigger, [34](#)
 - HasTrigger, [34](#)
 - MakeDrawable, [34](#)
 - Sprite, [34](#)
 - Tile, [33](#)
- AthenaEngine::Framework::Interfaces::ICollidable< T >
 - CollidesWith, [26](#)
- AthenaEngine::Framework::Interfaces::IDrawable
 - Draw, [26](#)
- AthenaEngine::Framework::Interfaces::IFocusable
 - Position, [27](#)
- AthenaEngine::Framework::Interfaces::IMoveable
 - CanMove, [27](#)
 - Move, [27](#)
- AthenaEngine::Framework::Primitives::BoundingBox2D
 - BoundingBox2D, [14](#)
 - Bounds, [14](#)
 - CollidesWith, [14](#)
 - Equals, [14](#)
- AthenaEngine::Framework::Primitives::CollidableEntity
 - Bounds, [20](#)
 - CanMove, [20](#)
 - CollidableEntity, [19](#)
 - CollidesWith, [20](#)
 - Level, [20](#)
 - Move, [20](#)
- AthenaEngine::Framework::Primitives::Directions
 - DOWN, [21](#)
 - LEFT, [21](#)
 - RIGHT, [21](#)
 - UP, [21](#)
- AthenaEngine::Framework::Primitives::DrawableEntity
 - Draw, [22](#)
 - DrawableEntity, [22](#)
 - level, [22](#)
 - Position, [22](#)
 - SpriteColor, [22](#)
 - SpriteController, [22](#)
 - SpriteSheet, [22](#)
 - SpriteSource, [22](#)
- AthenaEngine::Framework::Primitives::Entity
 - Height, [23](#)
 - Move, [23](#)
 - Position, [23](#)
 - Rectangle, [23](#)
 - Size, [23](#)
 - Width, [23](#)
 - X, [24](#)
 - Y, [24](#)
- AthenaEngine::Framework::Systems::LevelLoaderXml-
::Sprite
 - Collides, [32](#)
 - Name, [32](#)
 - Sprite, [32](#)
 - X, [32](#)
 - Y, [32](#)
- AthenaEngine::Framework::Systems::ResourceManager<
T >
 - Add, [31](#)
 - Get, [31](#)
 - ResourceManager, [31](#)
- AthenaEngine::Framework::Systems::Triggers


- encounter, [35](#)
 - test, [35](#)
- AthenaEngine::Framework::UI::UI
 - AddButton, [36](#)
 - Draw, [36](#)
 - Level, [36](#)
 - UI, [35](#)
- AthenaEngine::Framework::UI::UIButton
 - Draw, [37](#)
 - Level, [37](#)
 - UIButton, [36](#)
- AthenaTest, [11](#)
- AthenaTest.Engine_Tests, [11](#)
- AthenaTest.Engine_Tests.Primitives_Testing, [11](#)
- AthenaTest.Engine_Tests.Primitives_Testing.Bounding-Box2DTest, [14](#)
- BoundingBox2D
 - AthenaEngine::Framework::Primitives::Bounding-Box2D, [14](#)
- Bounds
 - AthenaEngine::Framework::Primitives::Bounding-Box2D, [14](#)
 - AthenaEngine::Framework::Primitives::Collidable-Entity, [20](#)
- Camera2D
 - AthenaEngine::Framework::Camera2D, [16](#)
- CanMove
 - AthenaEngine::Framework::Interfaces::IMoveable, [27](#)
 - AthenaEngine::Framework::Primitives::Collidable-Entity, [20](#)
- Character
 - AthenaEngine::Framework::Gameplay::Character, [18](#)
- CollidableEntity
 - AthenaEngine::Framework::Primitives::Collidable-Entity, [19](#)
- Collides
 - AthenaEngine::Framework::Gameplay::Tile, [34](#)
 - AthenaEngine::Framework::Systems::LevelLoader-Xml::Sprite, [32](#)
- CollidesWith
 - AthenaEngine::Framework::Interfaces::ICollidable< T >, [26](#)
 - AthenaEngine::Framework::Primitives::Bounding-Box2D, [14](#)
 - AthenaEngine::Framework::Primitives::Collidable-Entity, [20](#)
- Collisions_WithEnvelopedRectangle_IsTrue
 - AthenaTest::Engine_Tests::Primitives_Testing::BoundingBox2DTest, [15](#)
- Collisions_WithItself_IsTrue
 - AthenaTest::Engine_Tests::Primitives_Testing::BoundingBox2DTest, [15](#)
- Collisions_WithRectangleOnBottom_IsFalse
 - AthenaTest::Engine_Tests::Primitives_Testing::BoundingBox2DTest, [15](#)
- Collisions_WithRectangleOnLeft_IsFalse
 - AthenaTest::Engine_Tests::Primitives_Testing::BoundingBox2DTest, [15](#)
- Collisions_WithRectangleOnRight_IsFalse
 - AthenaTest::Engine_Tests::Primitives_Testing::BoundingBox2DTest, [15](#)
- Collisions_WithRectangleOnTop_IsFalse
 - AthenaTest::Engine_Tests::Primitives_Testing::BoundingBox2DTest, [15](#)
- DOWN
 - AthenaEngine::Framework::Primitives::Directions, [21](#)
- Draw
 - Athena::Game1, [25](#)
 - AthenaEngine::Framework::Gameplay::Level, [30](#)
 - AthenaEngine::Framework::Gameplay::Tile, [34](#)
 - AthenaEngine::Framework::Interfaces::IDrawable, [26](#)
 - AthenaEngine::Framework::Primitives::Drawable-Entity, [22](#)
 - AthenaEngine::Framework::UI::UI, [36](#)
 - AthenaEngine::Framework::UI::UIButton, [37](#)
- DrawableEntity
 - AthenaEngine::Framework::Primitives::Drawable-Entity, [22](#)
- encounter
 - AthenaEngine::Framework::Systems::Triggers, [35](#)
- Equality_WithIdenticalRectangle_IsEqual
 - AthenaTest::Engine_Tests::Primitives_Testing::BoundingBox2DTest, [15](#)
- Equality_WithItself_IsEqual
 - AthenaTest::Engine_Tests::Primitives_Testing::BoundingBox2DTest, [15](#)
- Equality_WithSameRectangle_IsEqual
 - AthenaTest::Engine_Tests::Primitives_Testing::BoundingBox2DTest, [15](#)
- Equals
 - AthenaEngine::Framework::Primitives::Bounding-Box2D, [14](#)
- FireTrigger
 - AthenaEngine::Framework::Gameplay::Tile, [34](#)
- Focus
 - AthenaEngine::Framework::Camera2D, [17](#)
- font
 - Athena::Game1, [25](#)
- Game1
 - Athena::Game1, [25](#)
- Get
 - AthenaEngine::Framework::Systems::Resource-Manager< T >, [31](#)
- HasTrigger
 - AthenaEngine::Framework::Gameplay::Tile, [34](#)
- Height
 - AthenaEngine::Framework::Primitives::Entity, [23](#)

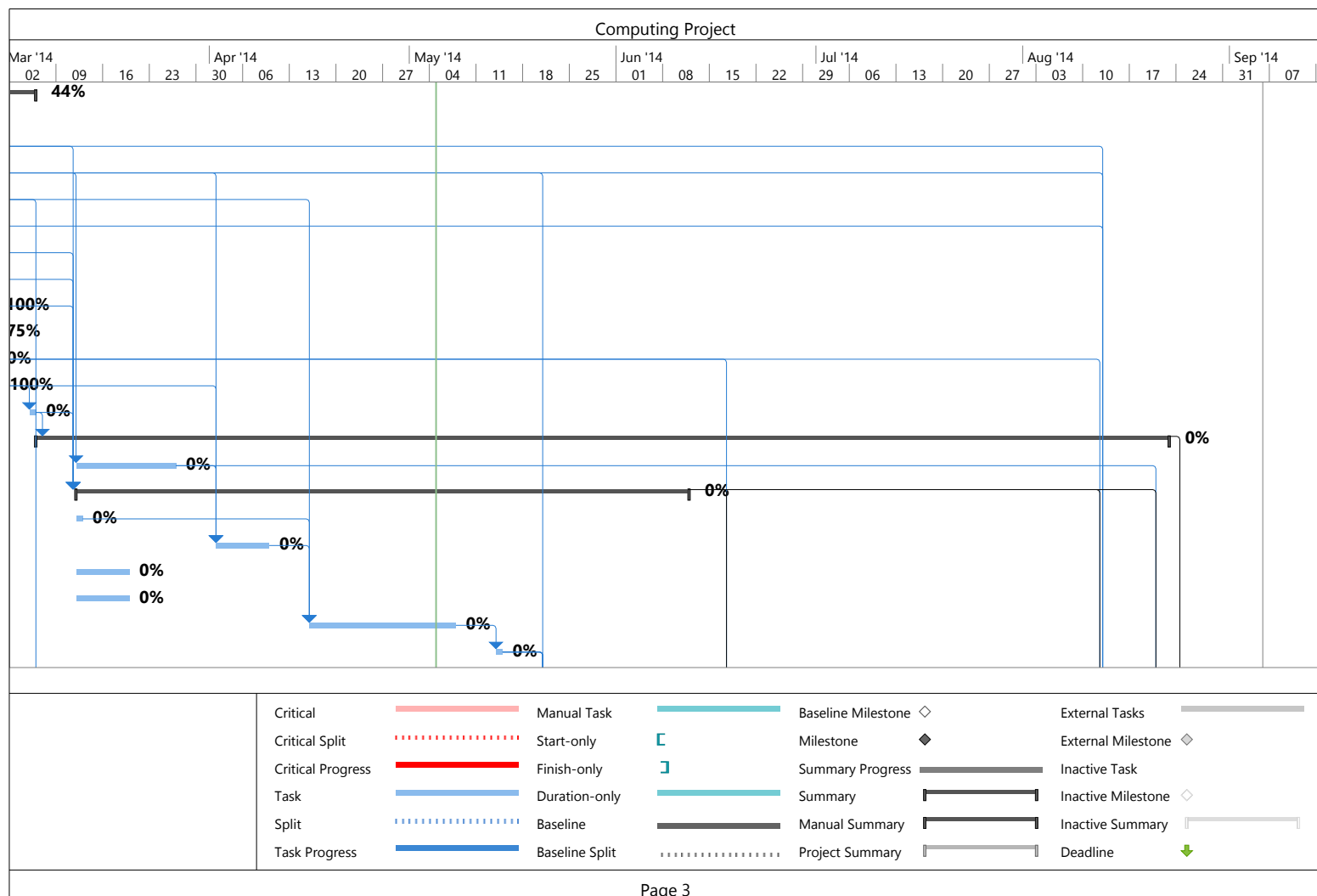
- Initialize
 - Athena::Game1, [25](#)
 - AthenaEngine::Framework::Camera2D, [17](#)
- IsInView
 - AthenaEngine::Framework::Camera2D, [17](#)
- Item
 - AthenaEngine::Framework::Gameplay::RPG::Item, [29](#)
- ItemInstance
 - AthenaEngine::Framework::Gameplay::RPG::Item-Instance, [29](#)
- LEFT
 - AthenaEngine::Framework::Primitives::Directions, [21](#)
- Level
 - AthenaEngine::Framework::Gameplay::Level, [30](#)
 - AthenaEngine::Framework::Primitives::Collidable-Entity, [20](#)
 - AthenaEngine::Framework::UI::UI, [36](#)
 - AthenaEngine::Framework::UI::UIButton, [37](#)
- level
 - AthenaEngine::Framework::Primitives::Drawable-Entity, [22](#)
- LoadContent
 - Athena::Game1, [25](#)
- MakeDrawable
 - AthenaEngine::Framework::Gameplay::Tile, [34](#)
- Move
 - AthenaEngine::Framework::Interfaces::IMoveable, [27](#)
 - AthenaEngine::Framework::Primitives::Collidable-Entity, [20](#)
 - AthenaEngine::Framework::Primitives::Entity, [23](#)
- MoveSpeed
 - AthenaEngine::Framework::Camera2D, [17](#)
- Name
 - AthenaEngine::Framework::Gameplay::RPG::Item, [29](#)
 - AthenaEngine::Framework::Systems::LevelLoader-Xml::Sprite, [32](#)
- Origin
 - AthenaEngine::Framework::Camera2D, [17](#)
- Position
 - AthenaEngine::Framework::Camera2D, [17](#)
 - AthenaEngine::Framework::Interfaces::IFocusable, [27](#)
 - AthenaEngine::Framework::Primitives::Drawable-Entity, [22](#)
 - AthenaEngine::Framework::Primitives::Entity, [23](#)
- RIGHT
 - AthenaEngine::Framework::Primitives::Directions, [21](#)
- Rectangle
 - AthenaEngine::Framework::Primitives::Entity, [23](#)
- ResourceManager
 - AthenaEngine::Framework::Systems::Resource-Manager< T >, [31](#)
- Rotation
 - AthenaEngine::Framework::Camera2D, [17](#)
- Scale
 - AthenaEngine::Framework::Camera2D, [17](#)
- ScreenCenter
 - AthenaEngine::Framework::Camera2D, [18](#)
- Size
 - AthenaEngine::Framework::Primitives::Entity, [23](#)
- Sprite
 - AthenaEngine::Framework::Gameplay::Tile, [34](#)
 - AthenaEngine::Framework::Systems::LevelLoader-Xml::Sprite, [32](#)
- SpriteColor
 - AthenaEngine::Framework::Primitives::Drawable-Entity, [22](#)
- SpriteController
 - AthenaEngine::Framework::Primitives::Drawable-Entity, [22](#)
- SpriteSheet
 - AthenaEngine::Framework::Primitives::Drawable-Entity, [22](#)
- SpriteSource
 - AthenaEngine::Framework::Primitives::Drawable-Entity, [22](#)
- test
 - AthenaEngine::Framework::Systems::Triggers, [35](#)
- Tile
 - AthenaEngine::Framework::Gameplay::Tile, [33](#)
- TileList
 - AthenaEngine::Framework::Gameplay::Level, [30](#)
- Transform
 - AthenaEngine::Framework::Camera2D, [18](#)
- UI
 - AthenaEngine::Framework::UI::UI, [35](#)
- UIButton
 - AthenaEngine::Framework::UI::UIButton, [36](#)
- UP
 - AthenaEngine::Framework::Primitives::Directions, [21](#)
- UnloadContent
 - Athena::Game1, [25](#)
- Update
 - Athena::Game1, [25](#)
 - AthenaEngine::Framework::Camera2D, [17](#)
- Width
 - AthenaEngine::Framework::Primitives::Entity, [23](#)
- X
 - AthenaEngine::Framework::Primitives::Entity, [24](#)
 - AthenaEngine::Framework::Systems::LevelLoader-Xml::Sprite, [32](#)
- Y

AthenaEngine::Framework::Primitives::Entity, [24](#)
AthenaEngine::Framework::Systems::LevelLoader-
 Xml::Sprite, [32](#)

10 Appendix 10 - Project File

Computing Project														
ID		WBS	Task Name	Duration	Start	Finish	Predecessors							
1		1	Planning	21 days	Wed 05/02/14	Wed 05/03/14		26	Feb '14	02	09	16	23	M
2	✓	1.1	Plan Genre/Setting	0.25 days	Wed 05/02/14	Wed 05/02/14								
3	✓	1.2	Plan Gameplay	0.25 days	Wed 05/02/14	Wed 05/02/14	2							
4		1.3	Plan Level Design	0.5 days	Wed 05/02/14	Wed 05/02/14	3							
5	✓	1.4	Plan Story	0.5 days	Wed 12/02/14	Wed 12/02/14	4							
6	✓	1.5	Design Characters	0.25 days	Wed 12/02/14	Wed 12/02/14	5							
7		1.6	Create Class Diagram	1 day	Wed 12/02/14	Wed 19/02/14	6							
8		1.7	Create Collision Diagram	0.25 days	Wed 19/02/14	Wed 19/02/14	7							
9	✓	1.8	Select Language/Framework	0.25 days	Wed 26/02/14	Wed 26/02/14	8							
10		1.9	Plan UI	0.25 days	Wed 26/02/14	Wed 26/02/14	9							
11		1.10	Plan Sounds	0.25 days	Wed 26/02/14	Wed 26/02/14	10							
12	✓	1.11	Plan RPG Elements	0.25 days	Wed 26/02/14	Wed 26/02/14	11							
13		1.12	Write Pseudocode	1 day	Wed 05/03/14	Wed 05/03/14	12							
14		2	Implementation	122 days	Thu 06/03/14	Fri 22/08/14	13							
15		2.1	Design Levels	3 days	Wed 12/03/14	Wed 26/03/14	4							
16		2.2	Program Game Engine	66 days	Wed 12/03/14	Wed 11/06/14	3,8,9,13,7							
17		2.2.1	Program Movement Logic	1 day	Wed 12/03/14	Wed 12/03/14								
18		2.2.2	Program RPG Elements	2 days	Wed 02/04/14	Wed 09/04/14	12,4,15							
19		2.2.3	Program Collision Detection	2 days	Wed 12/03/14	Wed 19/03/14								
20		2.2.4	Program Graphics Handling	2 days	Wed 12/03/14	Wed 19/03/14								
21		2.2.5	Program Story Handling	4 days	Wed 16/04/14	Wed 07/05/14	5,17,18							
22		2.2.6	Program Handling Of Sprite Sheets	1 day	Wed 14/05/14	Wed 14/05/14	21							
			Critical		Manual Task		Baseline Milestone		External Tasks					
			Critical Split		Start-only		Milestone		External Milestone					
			Critical Progress		Finish-only		Summary Progress		Inactive Task					
			Task		Duration-only		Summary		Inactive Milestone					
			Split		Baseline		Manual Summary		Inactive Summary					
			Task Progress		Baseline Split		Project Summary		Deadline					
Page 1														

Computing Project													
ID		WBS	Task Name	Duration	Start	Finish	Predecessors	Feb '14					M
23		2.2.7	Program Level Handling	4 days	Wed 21/05/14	Wed 11/06/14	4,22						
24		2.2.8	Integrate Random Encounters	2 days	Wed 21/05/14	Wed 28/05/14	22						
25		2.2.9	Program Fighting	2 days	Wed 21/05/14	Wed 28/05/14	22						
26		2.2.10	Program Sounds Handling	2 days	Wed 21/05/14	Wed 28/05/14	22						
27		2.3	Create Sounds	36 days	Wed 18/06/14	Wed 06/08/14	11,16						
28		2.3.1	Create Voice Acting	4 days	Wed 18/06/14	Wed 09/07/14	36						
29		2.3.2	Create Battle Sounds	4 days	Wed 18/06/14	Wed 09/07/14							
30		2.3.3	Create Environment Sounds	4 days	Wed 18/06/14	Wed 09/07/14							
31		2.3.4	Create Background Music	8 days	Wed 18/06/14	Wed 06/08/14							
32		2.4	Design Sprites	6 days	Wed 13/08/14	Wed 20/08/14	11,16,27						
33		2.4.1	Create Environment Sprites	2 days	Wed 13/08/14	Wed 20/08/14	4						
34		2.4.2	Create UI	2 days	Wed 13/08/14	Wed 20/08/14	3						
35		2.4.3	Create Character Sprites	2 days	Wed 13/08/14	Wed 20/08/14	6						
36		2.5	Write Game Script	2 days	Thu 06/03/14	Fri 07/03/14	5						
37		2.6	Combine Game Elements	2 days	Thu 21/08/14	Fri 22/08/14	16,27,32,15						
38		3	Evaluation	1 day	Mon 25/08/14	Mon 25/08/14	14						
39		3.1	Individual Critical Appraisal	1 day	Mon 25/08/14	Mon 25/08/14							
40		4	Project Hand In	0 days	Fri 05/09/14	Fri 05/09/14	38						



;