Telearena

Contents

[Document Version 3](#_Toc329724382)

[Logline 3](#_Toc329724383)

[Game Framework 3](#_Toc329724384)

[Audience 3](#_Toc329724385)

[Platform 3](#_Toc329724386)

[Proposed Development environment 3](#_Toc329724387)

[External references 5](#_Toc329724388)

[Dramatic 7](#_Toc329724389)

[Gameplay Description 7](#_Toc329724390)

[Camera and perspective: 7](#_Toc329724391)

[Ideology: 7](#_Toc329724392)

[User Interface 8](#_Toc329724393)

[Objective 8](#_Toc329724394)

[Game mechanics/Procedures 8](#_Toc329724395)

[Resources 9](#_Toc329724396)

[Control definitions 9](#_Toc329724397)

[Component list 11](#_Toc329724398)

[Guns 11](#_Toc329724399)

[Personal teleport gun 11](#_Toc329724400)

[Object teleport gun 11](#_Toc329724401)

[Object Teleport beacon gun 11](#_Toc329724402)

[Flying Camera gun 11](#_Toc329724403)

[Jumper platform gun 12](#_Toc329724404)

[Helpers 12](#_Toc329724405)

[Camera 12](#_Toc329724406)

[Teleport location beacon 13](#_Toc329724407)

[Teleport marker beacon 13](#_Toc329724408)

[Jump platform 13](#_Toc329724409)

[Generic Objects 14](#_Toc329724410)

[World static walls and decoration 14](#_Toc329724411)

[Teleport-able objects 14](#_Toc329724412)

[Buttons & Triggers 15](#_Toc329724413)

[Spawning Points 15](#_Toc329724414)

[Ammo boxes 15](#_Toc329724415)

[Environmental Elements 15](#_Toc329724416)

[Moving Walkway 15](#_Toc329724417)

[Wind Currents 15](#_Toc329724418)

[Acid/water 15](#_Toc329724419)

[Black Holes 15](#_Toc329724420)

[Traps 16](#_Toc329724421)

[Player stats 16](#_Toc329724422)

[Projectile base speed/velocity 16](#_Toc329724423)

[Gun Resources 16](#_Toc329724424)

[Health 16](#_Toc329724425)

[Player Speed 16](#_Toc329724426)

[Appendix A: Particular traps. 17](#_Toc329724427)

[Pendulums 17](#_Toc329724428)

[Trap Doors 17](#_Toc329724429)

[Spikes 17](#_Toc329724430)

[Moving walls 18](#_Toc329724431)

[Laser Grids 18](#_Toc329724432)

[Robots 18](#_Toc329724433)

[Turrets 18](#_Toc329724434)

# Document Version

* Version 1.0 🡪 Andres Ramirez E
  + Set up of the basic gameplay mechanics and description
  + Set up of the first iteration of the component list
  + Set up of the first iteration of the resource list
  + Set up of the control definitions
  + Left a couple of notes regarding mostly how to handle ammunition and
* Version 1.1 🡪 Erin Reynolds
  + Corrected Spelling
  + Added options for the engines and platforms
  + Added comments with suggestions on components to add
  + Added comments with solutions for certain design issues
* Version 1.2 🡪 Andres Ramirez E
  + Incorporated several comments from 1.1
  + Added suggested components Moving Walkways and Wind Currents
  + Clarified the use of the ammunition as several possible resources.
  + Clarified time management and display to the player.
* Version 1.2.1 🡪 Andres Ramirez E
  + Fixed a miss-reference on the GUI section, it should point now to the right components for this game.
* Version 1.2.2 🡪 Andres Ramirez E
  + Pending I will add this latter lol ☺

# Logline

Race against the clock or other players in a puzzle solving competition by either teleporting yourself or other objects around the “race track”.

# Game Framework

## Audience

The target audience is people from 14+ that find entertainment in puzzle like games with strategic components.

## Platform

* Main platform: PC/Mac (Taking advantage of Unity3D).
* Secondary platform: Web, browser games (maybe components of social games).
* Tertiary platform: Mobile – with emphasis on tablet gaming (Taking advantage of Unity3D)

## Proposed Development environment

* Main: Unity3D.
* Secondary: Unreal Engine.
* Tertiary: Source Engine.

## External references

1. Portal & Portal 2: This is a game that allows the player to create two ends of a wormhole through which he/she can go through and travel through the different levels. Each level provides is presented as a puzzle that the player has to solve by using the portal mechanics and the elements that are around him (such as cubes, buttons triggers etc.).

The portals tend to keep the physics going so that the inertia the player or the objects have is kept when traveling through the portal.

**Key Elements:**

* + Teleporting.
  + Puzzle solving.
  + Use of external elements to help you advance through the level.





*To the left is an example of a portal level, this is an example on how they build their puzzles for the player, on the right is an example of the two ends of the wormhole connected. (http://2.bp.blogspot.com/-qOpbeef5SCU/Tgh6qYICyRI/AAAAAAAABJI/huaUUzHCcRI/s1600/portal%2B2%2Bscreenshot%2B2.jpg and http://cache.gawkerassets.com/assets/images/9/2011/11/482a4f08298d9281fc877eb4592fddb4.jpg)*

1. Unreal Tournament: On this game the players compete for their own survival or that of their group in what resembles to a modern/future Sci-Fi roman arena. This game has a particular weapon called the translocator (<http://unreal.standardof.net/unreal-tournament-2004/weapons-and-tactics/translocator/>) that is really similar to one of the core components of this game project.

**Key Elements**:

* + Teleporting gun.
  + The idea of competence and arena like games.



*This is an example of the scenarios that appear in the Unreal Tournament game, the idea is to keep a sense of competence between the players. (http://pnmedia.gamespy.com/planetunreal.gamespy.com/newsimages/ut3hq/large/action/action20.jpg)*

# Dramatic

The game will position the player into a futuristic arena where he competes against other players and the environment to reach his final goal.

The arena will act as a holo-deck system where multiple environments can coexist without impacting the development of the game. In that sense, we might very well have scenarios that take place in a haunted mansion, a medieval castle, a spaceship or a laboratory; all of them will be just recreations of those scenarios inside the arena.

The idea is to keep the story away from the player much as what happens with the Unreal Tournament games, have the player be a random “gladiator” in this new age matches.

That being said the player will take the roll of a just initiated “gladiator” in this futuristic matches, he will first confront easy challenges and as the matches go on and on he will be presented with more difficult scenarios each time.

As the game progress, the player will get the approval of the crowd and will eventually become the arena champion.

# Gameplay Description

## Camera and perspective:

The player will be using a FPS (first person shooter) view camera as his main view, and will have access to a secondary FPS view in a smaller viewport for his secondary camera.

## Ideology:

The purpose of the game is to engage the player into a competitive environment but also to make him think about what he/she should do to solve the different puzzles that are in the environment.

In order to have a more lasting appeal and add replay value to the game we should add certain characteristics to the system. These characteristics will be organized into three tiers that will be released as DLC components for the game (with the exception of the first one which should come out when the game launches).

* Tier1: Some of the traps and environment components that affect gameplay will be dynamically created either by total randomization or by randomizing their selection from a pool of objects. As it is difficult to completely randomize we will make more use of the second technique.
* Tier 2: We plan to develop a simple level editor so that the community can add scenarios to the game, therefore promoting a sense of ownership, competition and challenge in the very community, and lowering the load on the developer side to constantly make scenarios.
* Tier 3: Structures and components in the scenario will be dynamically created on each match (similar to the Diablo system) so that each match is always different.

We will keep a simple art style that allows us to repaint and retexture it to adapt it to the different environments we want to create. That way we won’t have constrains on the kind of art we want to use but we will rather work on a per scenario basis.

## User Interface

The GUI will have two main components: Direct information and indirect information.

1. Direct Information: This is numbers pretty much. In general this information represents any kind of data that is not “natural”, that requires some sort of descriptive information. Elements that compose this kind of data are:
   1. Life bar.
   2. Weapons selection.
   3. Floating camera.
   4. Ammunition or resource if used.
2. Indirect Information: This is the information that can be gathered from environmental elements and that does not require a constant display on the GUI framework. Elements that compose this data are:
   1. Scenario components i.e. walls, terrain and decoration.
   2. Intractable objects for instance teleportable objects.
   3. Beacons and gun representations

# Objective

The player objective is to reach the final goal in a scenario before either the time runs out or any of the other players reaches the goal.

# Game mechanics/Procedures

* The player uses an array of guns that grants him/her different abilities depending on the gun that is being used. The player will trigger an animation when he changes guns, shots guns or activates guns. Also the art of each gun should be different enough to make a clear distinction from one another; the beacons will also be different one from the other so that the player knows what beacon does what.
* The main mechanic is teleporting, the idea is that the player can throw beacons that are affected by physics, and then teleport itself or other objects to the beacon location.
* There will be some elements that complement the core mechanic for example the player will be able to set up flying cameras to see positions that are difficult to access and to plan ahead how to solve certain puzzles.
* The scenarios will play on closed spaces and will present challenges to players such as 3D platform jumping, moving objects to reach certain spaces, or activating switches to get certain reactions.
* Death will be handled depending on the game mode being played:
  + Multiplayer 🡪 Use spawning point to resurrect, the player has to wait for a time before resurrecting.
  + Single player 🡪 The player passes through check points that will return the game to the previous state before the death of the player.
  + No Checkpoints 🡪 The player will have to restart the level from zero.
* The player will try to beat the clock or other players on the race to achieve victory.
* Guns will use resources to either shot or be activated, the resources may change per environment, or we will select a particular resource depending of playtest, we might even use multiple resources at the same time. The resources are:
  + None 🡪 No resource is used by the gun.
  + Cool down 🡪 The gun will take some time to shot or activate again.
  + Energy 🡪 Shooting consumes energy, and energy is regained over time.
  + Ammunition 🡪 Shooting consumes ammo, the player needs to find ammo crates to replenish lost ammo.

# Resources

* Time: Time is one of the main game resources, be it because the player is playing against the clock, or against other players. Time will be displayed to the player with a small clock in the HUD. Other elements may help the player know its current time such as rising acid pools or other environmental elements.
* Beacons: This is considered a resource since there can only be a beacon per weapon on the world at any given time. This constricts the player as they will have to chain actions to achieve an objective.
* Guns resources: Guns may use different kinds and combinations of resources, the final combination will be determined by play testing:
  + None 🡪 No resource is used by the gun.
  + Cool down 🡪 The gun will take some time to shot or activate again.
  + Energy 🡪 Shooting consumes energy, and energy is regained over time.
  + Ammunition 🡪 Shooting consumes ammo, the player needs to find ammo crates to replenish lost ammo.

# Control definitions

* Movement: movement will be defined as Axis X and Y. The movement axis will depend on the control that is being used, Y axis moves forward backwards, X axis moves left or right.
  + Example: on the keyboard the W and S keys represent the Y axis, the A and D keys represent the X axis, this leaves the player with the typical WASD movement system.
* Looking: looking will be managed by a secondary Axis system, where the Y axis will manage looking up or down, and the X axis will manage looking left or right.
  + Example: on a PC when using the mouse, the mouse movements will be mapped to the looking axis system (just as any FPS works).
* Jumping: The jump key will effectuate a small jump in the character.
* Shooting: The shoot describes the process of launching the gun beacons (beacons depend on the gun that is being used), and will be performed **once** every time the designated shoot key is pressed.
  + Example: On the PC the shooting key is the left mouse click.
* Activating: The activation procedure describes the effect that happens once the activation key is pressed. Activation can only happen **once per active beacon**.
  + Example: On the PC the activation key is the right mouse click.
* Weapon exchange system: This system will control what is the active weapon. Players are supposed to start with all available weapons and they can use them in whatever way they please.
  + Example: On the pc rotating the mouse wheel or pressing the numbers on the keyboard should reflect in a weapon change.

# Component list

## Guns

### Personal teleport gun

This is the main gun the player can use; its function is to throw *“Location Teleport Beacons”* and teleport the player to those beacons. The gun functionality is described as follow:

* Fire button: Launches a *“Location Teleport Beacon”* with a relatively slow velocity. The beacon will be affected by physics. Only one beacon of this gun can be in the scene.
* Activation button: Transports the player to the closest valid location to the beacon. A location is valid if the player physical representation can be in the room.

### Object teleport gun

This gun is used to teleport objects that have been marked with an *“Object Teleport Beacon”* to the position of a “Location Teleport Beacon”. The gun functionality is described as follow:

* Fire button: Launches a *“Location Teleport Beacon”* with a relatively slow velocity. The beacon will be affected by physics. Only one beacon of this gun can be in the scene.
* Activation button: Transports a tagged object (object that has an *“Object Teleport Beacon”* attached) to the closest valid location to the beacon. A location is valid if the player physical representation can be in the room.

### Object Teleport beacon gun

This gun is used to place *“Object Teleport Beacons”* on objects in the scene. It should be noted that not all the objects in the scene are subject to be teleported, for instance the ground or the walls cannot be teleported; however, crates and the like can be.

* Fire button: Launches an *“Object Teleport Beacon”* with a moderate velocity. The beacon will be affected by physics. And will attach to the first object that is subject to teleportation that it touches. Only one beacon of this gun can be in the scene.
* Activation button: Eliminates any current *“Object Teleport Beacon”* that is in the scene for that player.

### Flying Camera gun

This gun is used to throw flying “*Cameras*” into tight spots or difficult access locations and see your actions from a different perspective.

* Fire button: Launches the flying “*Camera”* in the direction, the camera will move for a short while by itself and then it will stay static until the player starts controlling it. Also, when activated, the player will get a small viewport on the lower right side of the screen that will show exactly what the camera sees.
* Activation button: When pressed the player will control the camera instead of his character, when pressed again he will stop controlling the camera and will regain control of the character.
* ESC button: The player will delete the camera and if in camera mode, he will regain control of the character, a message should be shown in order to alert the player of this option.
* When the camera is deleted the viewport should be closed (with an animation)

### Jumper platform gun

This gun will launch a platform on the floor (or on the walls) and when activated it will apply a force to any player in a direction normal to the platform.

* Fire button: When pressed the gun will launch a beacon than when hits the floor or a wall, creates a platform in its location parallel to the surface that was hit.
* Activation button: When activated the platform will start to apply a force in a direction normal to its surface, to any player that is nearby.

## Helpers

### Camera

1. The main camera is a FPS type of camera that is the representation of the main player. In general there should be a crosshair in the center of the screen that allows the player to point where they want to shoot.

The camera will use an effect to resemble a helmet much as it is shown in the halo games.

Example:



*Note how the screen has a surrounding image around it; that is the desired effect in the main camera. Source: BUNGIE Halo 3, web link http://www.bungie.net/images/Games/Halo3/Screenshots/H3\_E307\_FP01.jpg*

1. The secondary camera is a flying camera that is launched by using the “*Flying Camera gun”*, this is also controlled as an FPS camera, but this one can be moved freely on xyz axis. Also the camera doesn’t require having a crosshair. The desired effect on this camera is that of a color image but with some noise to represent static.

Example:



*Note how the screen has a granular effect that looks similar to noise in a texture, or to static in an old TV. Source: Capcom Clock Tower 3, web link http://videogamecritic.net/images/ps2/clock\_tower\_3.jpg*

### Teleport location beacon

The teleport location beacon fulfills two possible roles:

1. It teleports the player to the closest location to the beacon. This is done when using the *“Personal teleport gun”*.
2. It teleports a designated object to the closest location to the beacon. A designated object is that which has attached a “*Teleport marker beacon”* on it. The effect can only happen when using the “*Object Teleport beacon gun”*.

Since both effects are different, there will be the necessity to do two different beacons that although act similarly, they also are completely different in their mechanic. Nevertheless, both codes can share some common ground.

### Teleport marker beacon

The teleport marker beacon will be attached to teleport able objects and only those objects that possess that beacon will be teleported. This beacon will be shot from the “*Object teleport beacon gun*”. When the beacon is shot, it will immediately attach to the first teleport able object that it finds, and will mark that object as the next target of a teleportation.

### Jump platform

When Shot, the jump platform will be launched as any other beacon, when activated, it will stop all movement, and will start to fall until it touches any non teleport able object, at this point, the beacon will assume the form of a platform and will attach to the closest surface of the object that will apply a force in the direction of the normal of the surface that it attaches too.



*We can see on this image an example of a platform that is attached to the floor, our platform would generate a force on the player or other physical elements in the direction of the yellow stream. Web link: http://www.mobygames.com/images/shots/l/363212-unreal-tournament-iii-windows-screenshot-jump-platform-s.jpg*

## Generic Objects

### World static walls and decoration

The world will be compromised by static objects and moving objects that will act as both decoration of the environment and components of the puzzles that will compose the level. The idea is that most of this objects are static, with no mechanics attached whatsoever; however, some objects will have simple mechanics attached to them, for example: Elevators, Doors, Movable platforms, etc. Some of this movable objects will interact with either triggers or buttons, but they will not be affected by the player in any other way.

* Fully static objects 🡪 light baking, no physics rigid bodies and occlusion culling.
* Dynamic Objects 🡪 Have basic reactions and scripts, have physics rigid bodies, may not be considered static for occlusion culling, may still be considered for light baking.

### Teleport-able objects

Teleport-able objects are objects that can be directly manipulated by the player, and that may have consequences in the game. These objects might be used by the player to solve puzzles, hamper the progress of another player, or to help the player achieve his goals in some way or another.

These objects can be separated in two categories based on their physics behavior:

* Can be Kinematic objects (i.e. no physics applies to them)
* Can be normal physical objects.

Further mechanics may be added to the objects depending on a desired behavior, but the idea is that they all share a common element which is the ability of the player to manipulate their behavior.

### Buttons & Triggers

The buttons and triggers will be used to apply events on the level. The buttons will only be activated by the player itself, the triggers can be activated by either the player or by elements in the game like the beacons or the teleport-able objects.

### Spawning Points

The player will be able to use this to reappear in the game when he dies on a multiplayer environment. They can also be used as checkpoints in a single player environment so that when combined with triggers, will allow the player to return to that game stage in case he losses.

### Ammo boxes

This would be used to restore the number of beacons available to the player in case that guns use ammunition. Ammo boxes may be used to increase the recovery speed of energy and cool down, thus diminishing the time for the cool down to finish.

## Environmental Elements

### Moving Walkway

A set of platforms that will be constantly moving into one direction, they will help force the player to think faster and calculate better his times or otherwise he won’t be able to keep up with the treadmill, or won’t be able to use the moving platforms to solve the puzzle.

### Wind Currents

Wind currents will affect the movements of the beacons by altering their movement direction or by applying a constant force in one single direction.

### Acid/water

This would act as zones where if the player steps into will receive damage until he dies. The zones themselves can be either moved or left static. For example, in a level where a player is competing against the clock, the game would use a constantly rising area of acid that will push the player to move forward.

### Black Holes

Black Holes generate an area that will attract elements that enter into it to the center of the area. This is done by applying a constant “force” to the objects. The applied force can be variable and as such the game will end up having several different effects; for instance a very strong force will make anything in the field to the center of it, but a slightly strong force will change the movement direction of the object.

The idea for now is that the black holes can only affect beacons, and normal physical objects.

### Traps

The traps will be activated by the player and can cause different effects:

1. Damage to the player: much as how the acid works, it would be a constant damage that would eventually kill the player.
2. Teleports the player away from the goal. The desired destination might also be in a dangerous position or in a situation that places the player in disadvantage.

## Player stats

### Projectile base speed/velocity

Each beacon or projectile that can be “shot” may have a different speed value than another, for instance the

* This element can change from gun to gun

### Gun Resources

The guns may use different kinds of resources as explained above. Each resource will have a corresponding representation in the HUD, for instance the ammunition will present a set of “bullets” that will be depleted as they are used, the energy will display an energy bar that changes its size depending on how often it is used, the cool down will be represented by a clock that will go dark when used and will start to refill until it returns to its natural color.

### Health

The player life will regenerate really fast if the player is unharmed; if harmed it will take damage and will have to wait for a time before starting to regenerate.

* Designated by a life bar in the HUD display.

### Player Speed

The player speed will be for the most part of the game a static value that won’t change much; we can leave the door open for buffs and effects on it though.

# Appendix A: Particular traps.

Following is a description of some of the possible traps that the game can include, we want to keep the setup of the traps rather generic so that levels can be easily built and that we can keep an organized set of classes for our trap system. Nevertheless, we can always leave the door open for particular trap implementations on certain levels.

That said, he idea is to keep most of the trap mechanics simple and changing only their presentation from scenario to scenario (depending of the atmosphere), while keeping the internal mechanics intact for them, thus in a medieval scenario we might have a trap that shoot arrows from a wall, and in a futuristic scenario we can have another trap that has a machinegun or a set of machineguns in the wall; nevertheless, both traps share the same basic mechanics.

Next we will present a proposed set of trap mechanics for our game, we will keep generic names for the mechanics, but it is important to keep in mind that the implementation may not graphically represent the exact definition of the name but rather the description of the mechanic:

Note: Trap damage will be calculated in a separate file and will be adjusted based on playtest, we will be simply describing behaviors here.

### Pendulums

When the trap is activated it will throw a pendulum (most likely partially blocking a hallway) that will move fast and will force the player to either use his teleport mechanics or coordinate his movement to go through the trap.

* Does initial damage to the player every time it touches it.

### Trap Doors

More like a trigger than an actual trap, when step on the trap doors will open and will allow the player to fall through them to either a pit with spikes (see spikes trap), or a “bottomless” pit or a maze. The trap door might remain open for some time or immediately close after the player falls through it.

### Spikes

Spikes can come in different versions and are probably the most generic and customizable trap in the setting. Spikes can be either attached to moving walls (see moving walls) or a floor, or can be arrows or bullets throw from a wall and what not. All in all the spikes represent the most generic behavior of a trap in our setting and thus are highly customizable depending of the desired effect.

* Spikes can do either initial damage only (a lonely set of arrows), or constant damage regulated by an internal cool down (for instance spikes on the floor or a constant volley of arrows).
* Spikes don’t need to be spikes, acid, water (why not) and fire have the exact same behavior than the spikes, just with different parameters of damage and cooldow.

### Moving walls

As the name implies, two walls that start moving to a point where they touch themselves, they will start to apply damage to the player if they are close enough, and will kill him at a certain point by dealing 100% life damage.

* Constant damage at some point.
* Instant kill when close enough.

### Laser Grids

This trap is mostly based on the Resident Evil original movie. We have a set of two anchor points that can move independently and when the trap is activated we create a fine line between both anchor points, if the player is touched by the line he receives a certain amount of damage periodically.

* We should be able to produce arrays of anchor points thus generating random webs of lasers on the screen.
* Damage is done constantly to the player when he stays on the laser.
* Reference video: <http://www.youtube.com/watch?v=fPLax6k7bvQ>
* We may have to review intellectual property for this one.

### Robots

Robots are mostly automatons that will follow a predetermined path on the level and that when finding a player will try to stay close to him dealing damage in the process.

* Robots may be constantly moving, or they may be activated by a trigger and shoot from a location to follow their path.
* This will require a basic AI for the robots so that they return to their paths when they lose track of the player.
* Robots track a player once the player gets in detection range of the robots.
* Some robots won’t follow the player but they will do damage when they cross him, sort of how the blades on Zelda work. Reference video: <http://www.youtube.com/watch?feature=player_detailpage&v=Qq-vF784460#t=119s>

### Turrets

Turrets are objects that will attack players when they are close by. Turrets as with spikes, can have several different damage configurations, but they usually follow these rules:

1. They will attack anything or any player in a certain area.
2. They will display an animation to show who they are targeting.
3. They will do bursts of damage with varying cool downs.

* Turrets can go from a machinegun turret, to a laser like turret (see NOD obelisk for reference)