# Advanced FPS Kit

## Team Based Game Modes

**By: Robert C. Fritzen (Phantom Games Development)**

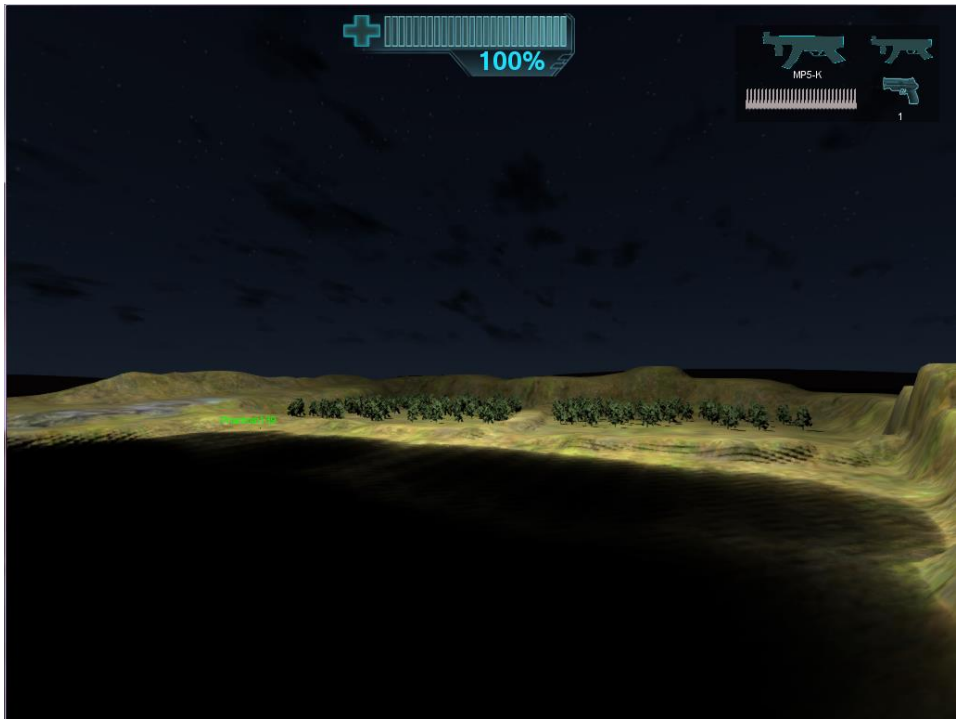
Welcome to my document / tutorial regarding the design and implementation of Team-Based Game Modes in multiplayer games, and for specifics, FPS games in general. Now, if you picked up my FPS Design pack when I released it a few weeks ago, you may recall how I mentioned that FPS Games are simply a combination of many different game modes masked over a generalized concept (Ok, not the exact terms I used, but you get the point), and today, I'm going to validate that prior statement to you by means of demonstrating how to set up and design a good team based game mode.

Now, what makes a team based game mode? Well obviously, the first term in there should stick out to you pretty quickly, it involves teams of players who work together to obtain a common goal, winning (Insert Charlie Sheen "Winning" Picture). Now, when it comes to designing a team based game mode we need to take into consideration some key factors in its definition (not just the group of devoted "must win" players). This is what a team based game mode by true definition is:

- A game where a group of players who work to accomplish a set primary objective together, upon completing the objective a score is added to a common team score which the final victory condition is judged on.
- A game where a set of secondary or subsidiary objectives may help the team score points towards victory, while not necessarily counting as much towards completing the primary objective.
- A game mode where each team member is responsible for either helping the team, or going a more mercenary (lone gun) route.

So, basically, the point here is that we'll be creating a mode where groups of players must work together to reach some common goal. Sounds rather easy, but in actual terms, it can be quite a difficult thing to do. Alright, so let's get right to it.

The first thing you'll want to establish right away is how large of a game mode is this going to be? One of the most important aspects of a game is establishing how many players will make up each team, and in certain cases, will completing objectives change this balance (IE: Elimination Games, Infection Games). Once you've got the idea, you need to check on your created maps, or to start conceptualizing the types of maps you want to use. The important part of a good team based game mode, is to ensure the flow of the gameplay fits the size of the map. You do not want to create your ultra-fast action "complete the game in a minute or less guaranteed" game mode that plays on the map named "Continent Wars Supreme Epic Oversize Edition"… just… don't…



This is not the map to play the fast paced game modes on

And then vice versa, you don't want to create a tactical team sniper shooter game mode in a recreation of Call of Duty's Scrapyard map (that ultra-small insta-death making map).

So, now that you've got that down, we can move onto the next part of the game mode, establishing the Primary Objective. This is where you specify what needs to be done as the condition for a team to claim victory, whichever team completes this objective first, or the fastest 'x' amount of times, wins the game. You can also define secondary objectives here that add to the team's score as well. While it may not contribute to winning the game, these objectives will add score to the team. In all team based game modes, eliminating enemy players is always a secondary objective. This is the step in the development where your creativity will enter the scene, and it's also where you'll want to create something new and different for your players to experience. It's also entirely acceptable to introduce game modes where there are multiple objectives that need to be completed in order to succeed, Bungie did this in Halo: Reach with its Invasion game mode and it was a huge success. Now, another key to remember here in the development process goes back to the FPS Design pack's rule #6, which states that a game that allows customization will be more successful than games without it. Let's discuss ways to design these concepts now.

First and foremost, when designing the task for your players, you should be thinking of a concept interface, think of how the game window will display for your players in the middle of a game. Which elements will need to be on the screen at any given point, and

how exactly are we going to present the scoring of the game to all of the players? Once you've got the idea of the game's gui down, you can picture at that point the objective of the game. By designing the gui for the game mode first, you can eliminate games that are way too complex in nature, and at the same time prevent it from ending too quickly. So now that we have the gui design down we can talk about actually implementing objectives that suit the game in general. There are a few types of common objectives to consider:

- Eliminating a certain player or players in general
- Capturing points on a map
- Capturing or destroying something belonging to the enemy
- Eliminating AI Players
- Surviving for the longest period of time

Each of these common objectives can have any number of subsidiary objectives or gameplay altering mechanics attached to them, for example you could create a special CTF mode where the player who grabs the flag becomes an overpowered suit of armor and then every other player must work to eliminate him. Keeping on this subject you need to remember that any game mechanics that are common to the entire game (such as weapon selection, killstreaks, ect) need to fit into the game balance, or if you're not a fan of balance and just want to unleash chaos, find a way to make the game objectives trigger unique events.


Or give the person who scores "God's AK-47"

For people making more complex game modes, or let's call them unique in nature for the purposes of this tutorial, your game modes will likely consist of completing multiple objectives and completing objectives will trigger in game events. For example let's use a game mode I'm working on for my shooter game, named Tunnel Escape. The game works fairly simple. You're trapped in a tunnel with 5 other players and need to escape, the problem is that there are debris blocking the route, and there are enemy AI approaching. A successful team will properly allocate team members to clear the debris, while sending others to eliminate AI, but here's the catch, each time a player dies, another AI spawns and players can only respawn at the checkpoints (when debris is cleared).

So, try to be inventive in nature when it comes to developing the objective of the game, you'll want to make the game do some extremely unique events while playing or completing objectives. Now let's talk about the next phase of making a game mode, which will be the gameplay itself. Now, FPS games have "core" gameplay mechanics which are things common to all games, such as your weapon selection, and special items (killstreaks, vehicles, ect), but your individual game modes while providing unique game styles, will also mess with these core mechanics.

The gameplay-objective relationship comes down to a few points of interest:

- How does completing an objective change mechanics (if they do)
- How does the game mode in general change the gameplay (Does it limit players to certain weapon loadouts, certain killstreaks, ect)
- Are there any special new mechanics added by this game mode?

So get your ideas down to these points of view, you'll want to nail down these questions pretty quickly when designing a game mode since the game mode itself will likely revolve around your ability to set up the code of the game mode, remember, the game will only do what you tell it to do so make sure that while your ideas are inventive in nature, not too overly complex that you won't be able to complete it. Now, let's jump back to my Tunnel Escape game mode and I'll run you through these questions:

In my game mode Tunnel Escape, the objective is simple, it's all in the name, you need to escape the tunnel. Each player spawns with a pistol of their selection and a special new weapon called the plasma cutter (used for clearing debris). You earn points by removing debris or by killing the attacking AI players. Now, completing a debris clearing will open up a "checkpoint" which will respawn any killed players, but the catch is doing so upgrades the enemy weapon tier to be more powerful, the players also now gain access to buy (with their points) weapons, upgrades, and killstreak superweapons. The players need to remain cautious however for two reasons, first and foremost, if they die, another AI is added (and the AI can freely respawn), so it makes the game harder for

everyone else, and secondly, this is still a competitive game mode, the player with the most points and the end wins.

So, revisit the three questions, it may seem like my paragraph above is just filled with the game concept in general, and while it is, you'll notice that I've answered all three questions right there. The objective is clearly listed, and I explain what happens when an objective is completed, I pretty well explained how the core mechanics of a general FPS game are altered by means of the limited weapon selection, and only access to choosing a starting pistol. I also answered the final question by exampling how costly death can be in my game mode. So, what you want to do is answer the three questions regarding mechanics adjustments in your game mode and then you can actually get to programming the game mode.

Now, going back to my FPS Design tutorial I'll help you get the basics for a team based game mode sorted out in the most systematically proper route. You'll want to start with a core game mode class (And thanks to T3D MIT, we already have one), and from there parent it into a sub class (Again, MIT handles nicely). So, the first task in a team based game mode is to assign the teams. Here is my code for handling team assignment; you can edit this to your liking:

```
function GameConnection::AssignTeam(%client, %game, %isTeamGame) {
   if(%isTeamGame) {
      // Get the count on both teams.
      for (%clientIndex = 0; %clientIndex < ClientGroup.getCount(); %clientIndex++) {
         %cl = ClientGroup.getObject(%clientIndex);
         %teamCount[%cl.team]++;
      }
      // Phantom: To-Do: Check for bot teams counts and add here as well.

      // Assign teams.
      if(%cl.team $= "") {
         //If the player hasn't already picked a team (via menu, we'll assign one)
         if(%teamCount[2] > %teamCount[1]) {
            %client.team = 1;
         }
         else {
            %client.team = 2;
         }
      }
   }
   else {
      for(%i = 1; %game.teamUsed[%i] != 0; %i++) {} //Run a simple loop to get the
last index.
      %game.teamUsed[%i] = 1;
      %client.team = %i;
   }
}
```

Now the next thing we need to do is set up the primary functions of the game, this

essentially is a “wrapper” class of the original game class, and it will use code of this format:

```
// ----------------------------------------------------------------------------
// MyGame
// FFA, 8 players fight to reach 30 kills within 15 minutes.
// ----------------------------------------------------------------------------

function MyGame::onMissionLoaded(%game) {
   $Server::MissionType = "MyGame";
   parent::onMissionLoaded(%game);
}

function MyGame::initGameVars(%game) {
   $Game::defaultPlayerClass = "Player";
   $Game::defaultPlayerDataBlock = "DefaultPlayerData";
   $Game::defaultPlayerSpawnGroups = "PlayerSpawnPoints PlayerDropPoints";

   $Game::defaultCameraClass = "Camera";
   $Game::defaultCameraDataBlock = "Observer";
   $Game::defaultCameraSpawnGroups = "CameraSpawnPoints PlayerSpawnPoints
PlayerDropPoints";

   // Set the gameplay parameters
   %game.duration = 15 * 60;
   %game.endgameScore = 30;
   //Post-game pause, 15 seconds
   %game.endgamePause = 15;
   %game.allowCycling = false;   // Is mission cycling allowed?

   for(%i = 1; %game.teamUsed[%i] == 1; %i++) {
      %game.teamUsed[%i] = 0;
   }
}

function MyGame::startGame(%game) {
   parent::startGame(%game);
   // Assign teams.
   for (%clientIndex = 0; %clientIndex < ClientGroup.getCount(); %clientIndex++) {
      %cl = ClientGroup.getObject(%clientIndex);
      %cl.AssignTeam(%game, false);
   }

   %game.getScores();
}

function MyGame::endGame(%game) {
   parent::endGame(%game);
}

function MyGame::cycleGame(%game) {
   //All the players hung around so give them their post-game reward
   %winScore = 0;
   %winClient = 0;
```

```
   for(%i = 0; %i < ClientGroup.getCount(); %i++) {
      %client = ClientGroup.getObject(%i);
      if(%client.score > %winScore) {
         %winScore = %client.score;
         %winClient = %client;
      }
   }
   parent::cycleGame(%game);
}

function MyGame::onGameDurationEnd(%game) {
   parent::onGameDurationEnd(%game);
}

function MyGame::preparePlayer(%game, %client) {
   //Pick a spawn-sphere from our list.
   %sphereGroup = FFADropPoints;
   for(%i = 0; %i < %sphereGroup.getCount(); %i++) {
      %sphere[%i] = %sphereGroup.getObject(%i);
   }

   %point = getRandom(0, %sphereGroup.getCount());
   %playerSpawnPoint = %sphere[%point];

   %game.spawnPlayer(%client, %playerSpawnPoint);
}

function MyGame::onClientLeaveGame(%game, %client) {
   parent::onClientLeaveGame(%game, %client);

   //Open the team this guy was on.
   %game.teamUsed[%client.team] = 0;
}

function MyGame::onClientEnterGame(%game, %client) {
   parent::onClientEnterGame(%game, %client);

   //Assign them to the next open team
   %client.AssignTeam(%game, false);
}

function MyGame::spawnPlayer(%game, %client, %spawnPoint, %noControl) {
   parent::spawnPlayer(%game, %client, %spawnPoint);

   //Fetch the player object.
   %player = %client.player;
   %player.setTeam(%client.team);
}

function MyGame::onDeath(%game, %client, %sourceObject, %sourceClient, %damageType,
%damLoc) {
   %client.player.setShapeName("");
   if (isObject(%client.camera) && isObject(%client.player)) {
      %client.camera.setMode("Corpse", %client.player);
      %client.setControlObject(%client.camera);
```

```
   }
   %client.player = 0;
   %sendMsgFunction = "sendMsgClientKilled_" @ %damageType;
   if (!isFunction(%sendMsgFunction)) {
      %sendMsgFunction = "sendMsgClientKilled_Default";
   }
   call(%sendMsgFunction, 'MsgClientKilled', %client, %sourceClient, %damLoc);

   if ((%damageType $= "Suicide" || %sourceClient == %client) &&
isObject(%sourceClient)) {
      %game.incDeaths(%client, 1, true);
      %game.incScore(%sourceClient, -1, false);
   }
   else {
      %game.incDeaths(%client, 1, false);
      %game.incScore(%sourceClient, 1, true);
      %game.incKills(%sourceClient, 1, false);

      if ($Game::EndGameScore > 0 && %sourceClient.score >= $Game::EndGameScore) {
         %game.cycleGame();
         return;
      }
   }
}
```

As you can see, the essential job of this class is to inherit the members of GameCore, our main game class and then modify the functioning of it to handle the events of our new game mode. At this point in time in development of your game (in general now) if you haven't done so, you will want to code in a few additional hooks in your code to check for game modes for purposes like team damage (friendly fire) and other factors like that, we want to make sure our team mates play along and by the rules of the game so make sure everything like that is set in stone and ready to go.

The last thing you should absolutely do after designing complex code like this is to get a bunch of your friends together and just play it, play the game like no tomorrow to ensure everything is running smoothly and that the gameplay is to your liking, and then as FPS Design states, balance the heck out of it to mess with it. The decision is yours and I leave you to it, thanks for reading! ☺