**Prototype Design Document**

04/11/2016

**High Level Design:**

1. Each Helmet Computer (HC from now on) runs a special program that talks over WiFi to other HCs. These programs are called “player characters” (PCs) when representing a real person's location (and maybe other stats eventually) and “non-player characters” (NPCs) when representing bots used to simulate real players. Initially we may not have any real PCs but the software architecture can be tested with only bots (NPCs).
2. System emphasizes quick response to dangerous events. Local picture of field around each PC is constructed from sensor data. Global picture is assembled from group of PCs. There are two possible models for how each PC senses the other players.
   1. Hub and Spoke. All PCs send their location information to one of the PCs (the MPC described in the above table). They then ask it for a picture of the field and location of all the other players. This might help synchronize all PCs to have the same view of the field but means that location information must make two “hops” to get from one PC to another. PC → MPC → PC.
   2. Distributed. Each PC exchanges message with all others. There is no one canonical version of reality, each PC may have a slightly different idea of what's out there. The advantage is that this is probably the most responsive solution.
   3. In practice a hybrid solution is likely with the MPC being used as a “referee” in cases where a PC has confusing or contradictory information about other PCs' status, location, or velocity.
3. Scalable and extensible design. The system is designed to accommodate the possibility that, for instance, multiple methods of ranging might be eventually needed to assure accurate ranging. One might combine WiFi, Bluetooth, and some other technology to arrive at accurate positional information. This would make the “Brain” component (see below) more complex and add to the number of sensor threads but will not require any major architectural changes.

**Design Breakdown:**

1. **SS**: Startup script. This is where it all starts. The script is launched from a sideline computer. The script needs to do the following. The steps follow one after another in time:
   1. Read global config file. CF contains definitions of PCs, NPCs, and MPC.
   2. Launch NPCs. These are bots that follow a path in space-time. Basically they're only there to test the system. There might be 10-20 of them. Each NPC is a separate process because eventually they will actually be real PCs running on different Pis.
   3. After the script has started human player(s) may join in and should be able to “see” all the bots around them. The GUI may also be started at this time. If all is working well, the GUI should display an image of the field with moving dots for players.
2. **PC / NPC**: This is the main software component. It has several readily identifiable components that do not rely (in time sequence at least) on each other. These kinds of parallel tasks are good candidates for multi-threaded development. The Raspberry Pi3 that we will probably end up using is now a quad core machine which should give us excellent multi-threaded performance. As a result I've broken down the PC into 5 essential parts, Brain, First Aide, Sensors, Post Office, and Shared Data Manager. When testing the system most of the players might be bots so there would be a fifth Bot thread. Here is what each thread does:
   1. ***Brain thread:*** First to start, spawns all others. Reads PC config file. CF determines whether we have a PC or NPC
      1. Makes connection with WR through SDM.
      2. Spawns Sensors (initially just the 3)
      3. Start processing Loop (continues for life of PC)
         * Processes data returned from sensors to compute its position.
         * Writes its position to WR through SDM or PO\*
         * Analyzes WR and its position/velocity within it. Mathematical trilateration is used to determine position and interpolation from previous positions is used to determine velocity. Positional error is propagated through the calculation so that we know how fuzzy the location data is.
         * Performs threat assessment.
           + If threat(s) detected, notifies FA (First Aide)
   2. ***FA (First Aide thread):*** Its job is to activate warning lights or sound effects to warn player of danger. It may do this in an interactive way at some point, perhaps monitoring player's response or lack thereof and stepping up the level of alert if need be. Brain thread tells it when to act. Potentially, though, the PO could directly address it when there are very critical messages (i.e. no need to get the brain involved).
   3. ***Sensor thread(s):*** Three of them are spawned by Brain when it starts to allow for mathematical trilateration of position.
      1. Acquire high precision WiFi signal strength data from one router/AP with high time resolution.
      2. Share signal strength data with parent thread (Brain).
   4. ***SDM thread:*** Shared Data Manager. Essentially this thread is in charge of assembling the WR for each PC from data gathered from other nodes. Its job is to prioritize data coming from other nearby players. It manages read/write locks to prevent concurrency issues if need be.
   5. ***Post Office thread:*** Handles inbound and outbound messages. This is used by all the other components of the system. Needs to support message queuing, different priority levels (i.e. emergency messages get bumped to the front of the queue even if they arrive after others), and timestamping of data, preserving all timestamps on data assembled from previous data. Old data should be less trusted than newer. In actual implementation this and the SDM may actually be in the same process thread if other player's location messages are sufficiently similar to other types to be handled the same way. PO may route certain messages directly to FA to sound critical alerts. This is somewhat borrowed from biology where certain sensory inputs directly trigger motor neurons that produce movement without the brain being involved.
   6. ***Bot thread:*** FOR NPCs only! This is a thread that simulates the data a sensor would receive as it moved around the field using the path created for it in the test scripts. Data is then fed to sensors which do not know that it is not real. Bot will track along a path specified in its config file but perhaps with some random jitter thrown in.
3. **MPC:** Special purpose PC that runs on a stationary computer on the sidelines. Can be computationally intensive if need be, perhaps even running on a cluster. For low end consumer use though this would need to be a low power machine only slightly more expensive than the Pis on the helmets. Perhaps a set of several Pis in parallel would work. MPC's job is to create a global authoritative picture for the coach/players and has “administrative” rights to alter the behavior of other PCs in the system. The MPC's version of reality is the fallback when a PC is confused or gets contradictory information in the hybrid method of WR.
4. **GUI:** OS-independent GUI view/controller (in the Model-View-Controller software paradigm). Allows coach or refs to get a “god's eye” view of the field and where all the players are. Can also show stats like average running speed/accelerations/estimated energy expenditure. This latter could very useful! Right now, estimating “how tired” someone might be is an inexact science. Knowing that a player has run the whole length of the field 10 times in the game while a different one has only done this 3 times might give a better idea of each player's remaining useful play time, especially when combined with injury and age data.
   1. If GUI is designed to be web based this could free the coaches/refs from needing to be at a particular machine as long as they have internet connectivity.
   2. This might also open up marketing possibilities with fans being offered live feeds of plays and stats on an interactive web page for a fee. The same GUI used by coaches and refs could be offered to fans but with the administrative authoritaty turned off.

**Estimates and Roadmap:**

**Stage 1 Prototype (Design Validation)**

This stage is purely to demonstrate that the software design works, that the PCs can track each other and that collisions can be predicted when imminent. At this stage we will not have an actual player or actual signal strength data feeding the sensors. That will be simulated. I'm leaving that out for several reasons. First of all, it might be necessary to write some low-level code (C or C++) to get signal strength data into the sensor threads (which will likely be written in Python or Ruby). There's also the possibility that each HC might need 3 WiFi dongles (one for each router) in order to get signal strength data at a high enough data rate. My initial idea was to scan for routers the way the wifi scanner programs do but I looked into it and I think that using a scan is much slower than an actual connection. From my previous network experience I also know that configuring multiple networks is always a bit of a mess. The system will function exactly the same once this is worked out and we can use simulated strength data in the meantime. **(see Prototype-Stage1 spreadsheet)**

**Stage 2 Prototype (Hardware/Software Integration)**

This is the stage where we start acquiring real data and try it on real hardware. Hardware/software integration becomes the main challenge here. Production of a few physical prototypes and testing with test subjects can start. **(see Prototype-Stage2 spreadsheet)**

**Stage 3 Prototype (Helmet Integration)**

At this point the hardware and software should be proven to work together. We can begin designing enclosures and fitting the devices to actual helmets. 3D printer can be used to create physical prototypes. **(No spreadsheet yet)**

**Terminology:**

| Term | Meaning | Description | Comment |
| --- | --- | --- | --- |
| PC | Player Character | These terms are borrowed from the video gaming world. |  |
| NPC | Non-Player Character | These terms are borrowed from the video gaming world. |  |
| MPC | Master Player Character (also a reference to Tron's MCP :) | Beefed up PC or NPC with administrative rights over all the others. Probably the coach's sideline field computer. | Need not be that mobile and thus can be more resource hungry than the PCs |
| SS | Startup script |  |  |
| HC | Helmet Computer | Raspberry Pi that each PC process runs on. | HC and PC are somewhat interchangeable terms but technically HC is the physical computer and PC is the program it runs. |
| WR | World Representation | Map of other players' locations, each PC maintains one. |  |