**To:** Professor Pisano, Professor Kia

**From:** Michael Abed, Jonathan Bell, Johan Mickos, Josh Navon, Mark O’Brien

**Team:** #15 The Karty Boyz

**Date:**  05/14/14

**Subject:** Functional Deliverable Test Plan

# **1.0 Overview**

## 1.1 Customer Requirements

The functional test aims to demonstrate the successful integration of all project deliverable and to emulate how a typical user would interact with the product. The full test will consist of:

* Uploading a pre-made test video of a Mario Kart 64 session
* Video processing software extracting individual races and uploading these to their corresponding Amazon S3 bucket
* Video processing software analyzing and extracting events from each race, populating the database with them
* Running unit tests to demonstrate successful deliverables
* Constructing a sample database query to demonstrate UI and query language

### 1.1.1 Event Detection

A significant component of the project is event detection. In Mario Kart 64 there is a great deal of events occurring every minute that define the current state of the race. The requirements for the types of events to be detected is as follows:

1. Changing places of characters
2. Beginning and end of a race
3. Lap changes
4. Acquiring items
5. Shortcut detection (specific to Koopa Troopa Beach map)
6. Exciting moment tagging based on volume levels
7. Key phrase recognition for manually tagging events
   1. The phrase “Tag it!” must be detected.
   2. False positives may be no more than 5% of the time.
   3. False negatives must be no more than 3% of the time.

Video-dependent events must be detected with at least 90% accuracy, with no more than 5% of the detected events being erroneous.

### 1.1.2 Database

A persistent database is required to store events, races, and sessions. These data must be accessible to general user queries. Specifically, the database must allow for storage of all types of events described in the previous section. Additionally, it must allow for users to ask for specific event queries without needing to use SQL.

### 1.1.3 Cloud Services

In addition to the database, a number of general computing tasks are required. These include video streaming, processing, and storage:

1. Video Services
   1. Users must be able to upload Mario Kart 64 race videos
   2. Users must be able to download and view race videos after processing
2. Video Processing
   1. Uploaded videos must be split into individual races and stored persistently. Race separation must be performed with 100% accuracy.
   2. Races must be processed for all events described above.
   3. Processing of a race must take no longer than twice the race duration.

### 1.1.4 Web Site

A website is needed to facilitate easy uploading, viewing, and querying of Mario Kart 64 data.

1. Session Uploads
   1. A user can upload a session video for processing
   2. Speed is dependent on their upload bandwidth
   3. Videos will be stored indefinitely
2. Race Downloads
   1. Once uploading a race, the user can view the individual races.
   2. User can see a list of videos to which he has access.
3. Event Querying
   1. The user can input queries on the website.
   2. The results from these queries can also be viewed on the website.
   3. The query language will be detailed on the web site.

### 1.1.5 Hardware

Microphone module is needed to capture external audio into session videos so that audio processing may be executed on it.

1. Microphone should meet following requirements.
   1. Amplify audio signal from electret microphone component.
   2. Output is such that it will not overload USB capture device.

## 1.2 Significance

Mario Kart 64 has been a favorite of many since it was released. Therein lies a problem, though. The life of the modern Mario Kart 64 gamer has turned into a dismal, meaningless existence. Never lacking worthy opponents, skilled players are left helpless, unable to stroke their egos as they fruitlessly insist upon their dominance.

Few have tackled the issue of gathering data and statistics from actual Nintendo 64 (N64) game-play, presenting us with an exciting opportunity. The Karty Boyz will make it possible to gather race and session statistics for Mario Kart 64, enabling players to back up their smack-talk with cold hard stats. Furthermore, this project aims to answer some of the mysteries of Mario Kart, such as, how random the item generation sequence is, to what extent does place affect the item received, the probability of winning when you miss the shortcut on Koopa Troopa Beach on lap 3, etc. Such information will prove useful to enthusiasts who can use it to improve their gameplay. With the ability to also tag events with speech, exciting moments will never be forgotten, providing an even more immersive experience. The finished product will give the devoted retro-gamer a clearer understanding of the game, a medium to better formulate strategies, a fun and interactive post-Mario-Kart experience, and a way to prove lofty claims of superiority.

# 2.0 Criteria and Process

## 2.1 Measurable Criteria

### 2.1.1 Video Processing

Successful completion of the functional test with respect to video processing will be determined by:

* AWS integration
  + The processing pipeline is driven by Amazon’s Web Services (AWS), with the Simple Queue Service (SQS) populating job queues for the virtual machines. Measurable criteria regarding the SQS queues will be the number of queue messages that get generated at each phase of the upload/processing sequence. For the pre-made test video, there needs to be one queue message generated in the *split-queue*, and every race within the test video needs to have a unique job in the *process-queue*.
  + Additionally, after the splitting of the test video has completed, the S3 bucket *race-videos* needs to contain separate videos for each race detected in the test video.
* Correct video splitting
  + The number of separate race videos generated in the first phase of video processing needs to exactly match the number of races in the test video.
* Accurate event detection
  + The unit tests for all detectors will indicate whether or not the event detection deliverable has been met. The measurable criteria for this is whether all unit tests pass.

### 2.1.2 Event Analytics

* Database
  + There will be three relevant tables stored in a database hosted by Amazon Relational Database Service (RDS). These are Sessions, Races, and Events.
  + The number of rows in the sessions table should correspond exactly to the number of session videos uploaded. The number of rows in the races table must correspond to the number of races in uploaded session videos. The number of rows in the events table must correspond to the number of events in the races.
* Query Language
  + To facilitate easy analysis of collected events a query language will be written.
  + A user should be able to put in a text based query that gets compiled to SQL that is equivalent to what a person writing SQL would write.
  + The query language should be able to express the following questions:
    - Which races were on Koopa Troopa Beach?
    - How many items has Toad ever gotten?
    - How many items did each finishing place get?
    - Which races did somebody get more than one Green Shell in?
    - What was the frequency of each type of item by place it was received in?

### 2.1.3 Website

* Upload
  + The user can upload a video
* Download
  + The user can view a race video
* Querying
  + The user can specify the query and view the results on the website

### 2.1.4 Audio Processing

* Functional hardware
  + Microphone module is able to capture external gameplay audio.
* Exciting Moments
  + Processing is able to extract exciting moments during gameplay and create corresponding events.

## 2.2 Testing Process

The testing process will consist of:

1. Recording a Mario Kart 64 with audio.
2. Uploading a pre-made Mario Kart 64 session video to the website
   1. We will verify that the AWS integration works correctly by monitoring the processes running on the virtual machine (VM) used for testing. Once uploaded, the code for phase 0 of processing will be launched on the VM.
   2. Additionally, we will be monitoring the AWS console to see SQS messages in-flight, representing dispatched jobs.
3. On a separate machine, we will run all unit tests for detection. This will be a visual representation of the accuracy of every detector. The full unit test will take roughly 10 minutes.
4. We will demonstrate the querying interface on the website by providing sample queries on pre-seeded data in the database.
5. To demonstrate how phase 0 of processing behaves, we will view race videos in the S3 bucket that have already been split from a previous session.
6. To show the ability to display a video, we will show the sample race.
7. We will show a short demo of how exciting moments in the races will be extracted.
8. Finally, as a temporary workaround to speech recognition, we will demonstrate our script which wraps Google’s speech api.

# 3.0 Equipment and Setup

## 3.1

*Describes equipment and setup in sufficient detail*

|  |  |
| --- | --- |
| Virtual machines | An instance within Amazon’s Elastic Compute Cloud (EC2) which will have all required libraries installed at the start of the test. |
| Client web browser | Used to upload the test session video to the cloud. |
| Pre-made session video | This will be uploaded to the website and trigger jobs to be dispatched. |
| Website | The website will be running at the start of test |
| USB Composite video capture card | Capturing video and audio from gameplay |
| Microphone module | Picking up external audio during gameplay |
| Game console | Nintendo 64 and controllers |

# 4.0 Conclusions

## 4.1 Assessment Process

After the functional test has completed, we will have the following data available to us:

* Console output of unit tests for detection indicating success/failure
* Console output of virtual machine processing races
* Session, race, and event data within the database
* Individual race videos in the S3 bucket after splitting
* Results of running queries written in our query language

In order to conclude the test as successful, we will compare these data to the expected outcomes. The database will contain an entry for the newly uploaded session, as well as entries for each individual race within the session. We will inspect the individual race entry to verify that the virtual machine populated the database with new events. The files created during the splitting process will be inspected to verify that they do indeed represent full races.