**BushBurg: Survival in Africa**

**Motivation for Pursuit**

It is easier than ever to feel connected to the world around us—at least, parts it. It is common for us to keep up with news in Europe, Asia, and the Middle East. Many of us have regular interaction with those an ocean apart through online communities. However, increased connectivity between inhabitants of the first world has further disconnected us from those of the third world. It is still difficult to make people care about those with which they have shared no contact. Additionally, the image of charity has taken something of a beating in recent years. Between fraudulent enterprises, counterproductive results, and the opacity of most charity expenditures, it has become difficult for charities to justify their existence to new generations. The pursuit of funding and volunteers for humanitarian efforts must evolve with the 21st century.

BushBurg is an attempt at exactly that type of evolution. Gaming is regarded as one of the most subversive and effective means to understand how it feels to be in another person’s shoes. The medium has also been shown to convince people to invest money willingly on a regular and long-term basis. These two facets of the industry lend themselves very well to raising money for charity. The trick, of course, is to make a compelling game experience which facilitates donation.

Our job is to build the foundations of that game experience, for a casual mobile market. The client wishes to simulate, in spirit, a plausible development of a community in rural Zambia. The cultivation of land, the perils of the environment, and the hope of investment paying off should all be represented within the game. There is no expectation of the struggles of the player to be true to life, but the idea is to develop a kinship with the simulated inhabitants. Done right, this process will motivate players to donate through in-game purchases; perhaps this will be done for self-interested rewards in game, or simply for the sake of helping. Those in need will benefit either way.

**Required Functionality**

The mobile marketplace for games is volatile and competitive, with failure often being absolute and success rare. Care must be taken to consider the nature of the market and what has been shown to work within it. The needs of the market and the needs of the client may not be inherently compatible, so compromises will be made. The following is a list of requirements imposed by the medium, the market, or the client, in that order.

* **Touch-Screen Interface (constraint of medium)**

Interaction with the game will typically be done on a phone or tablet device. This means that screen real-estate will be at a premium, and that precision input cannot be expected. The ‘swipe and drag’ functionality of touch-screens does provide a few additional design options.

* **Low session time requirements (constraint of medium/market)**

Whether it can be connected to battery-life concerns or the inherent ‘short break’ nature of phone interaction, we must be careful not to require playtime sessions in excess of 3-5 minutes at a time.

* **Free access to play (requirement of market)**

Even well-reviewed titles with costs of entry almost universally perform poorly on mobile markets. This means that the monetization model of the game will be what is dubbed “freemium,” with small transactions offered to enhance the player’s experience throughout. This has extreme implications for the mechanical design of the game, and will be probably the largest design challenge.

* **Motivation for long-term investment (requirement of market)**

Free-to-play titles get much more return from players that have been involved with the game for long periods, often in excess three months. The game must be designed such that it is continuous and cannot be completed quickly.

* **Slow increase in complexity (requirement of market)**

The game should not overwhelm new players with complicated systems. Since there was no initial cost, players which are unable to understand the game immediately will simply uninstall it. However, given the previous long-term investment requirement, the complexity does need to exist at some point to keep players interested in improvement. This simply requires a careful approach to the introduction of new complexity.

* **Perception of fairness** **(requirement of market)**

All items or currencies available for purchase for real money must be otherwise attainable. If this is not the case, the game will likely be written off as predatory by users. Generally this balance manifests in terms of time investment vs money investment.

* **Bias toward success (requirement of client)**

The game, while respecting the challenges of daily life in rural Africa, is expected to emphasize the positivity and hope of its residents. As such, it should not be punishingly difficult, and will have a ‘little victory/little failure’ ratio of roughly 80/20. Management of this ratio can be difficult for a simulation, and will require testing.

* **Community focus (requirement of client, planned but not implemented)**

While the implementation of a game community is out of scope for the project, the prototype must be conducive to community interaction and cooperation. This could be through trading systems, a sub-community system, and cooperative donation strategies.

* **Real-world tracking of donations (requirement of client, planned but not implemented)**

The game should lend itself toward active involvement in the donation process, through which the player will be made aware of their influence on real world events. For the prototype, this means that in-game donation results should have real world analogs.

**Implementation Overview**

The options for platform are quite few, as most game engines have significant time investments involved. This project was taken on in the first place because there is a viable option in the Unity engine. Unity is a multi-platform engine and editor that is capable of making mobile titles. This editor removes any required implementation of things like physics, rendering, or data structures for game objects. Since those systems are so difficult to develop, having a rigorously tested engine available to begin with will free us to spend our time with the actual game design instead. The Unity engine is not free for commercial enterprises, though it is certainly one of the least expensive options in that regard. It is, however, free for our purposes in creating a prototype. Licensing only matters when the product is launched.

The Unity engine supports primarily C# and Javascript in its scripting. C# has good documentation, widespread use on this platform, and both members of the team have some experience with it. While C# is garbage-collected and often not efficient enough for very performant titles, our project will not test the limits of modern phone processors. As a result, it has been chosen for use in the prototype.

Implementation details for the requirements will be discussed further in the project plan. A rough set of guidelines for each requirement follows.

* **Touch-screen interface** – Careful considerations will be made throughout to ensure selection and commands can be entered without precision. Limited persistent UI will ensure enough space to see what’s going on.
* **Low session** **time requirements** – Decisions made by the player will be quick to process but take time for the simulation to execute. This means that the player may interact with the game for a few minutes and then leave it to run for an hour or even longer.
* **Free access to play –** This mostly will serve as an overall design guideline for the mechanical balance of the game. Players can do anything within the game, but some actions may take a long time or require skill on the part of the player. Both of these pressures on the player can be alleviated by in game purchases.
* **Motivation for long term investment –** The simulation will be designed to develop over the course of months, with progress checkpoints that represent a limited victory. Subtle influences toward donation will increase after a certain period.
* **Slow increase in complexity** – The first few checkpoints of progress will be quick and easy to accomplish. The basic systems (which will be designed first) should not test the player’s limits, but simply acclimatize them to the goals of the game.
* **Perception of fairness** - This is not likely to be within scope, as it is a monetization strategy. However, we will develop some guidance to the client on this matter.
* **Bias toward success** – For the prototype, the player will not initially have a failure state. If time allows, there may be implementation of character death, but not total loss of a village. The client will be consulted as time goes on in this regard.

**Technical Challenges**

Use of an existing engine absolves us of the more prohibitive complexities involved in the design of a video game. Additionally, we will not be responsible for aesthetic concerns within the prototype, which will allow us to focus purely on the mechanical underpinnings of the game itself. To that end, the programming will initially not be taxing, especially with some experience in the field. However, two aspects of this project in particular will pose considerable technical challenge.

The first of these is the simulation artificial intelligence. Since the game must not impose constant micromanagement by the player, many decisions will have to be made by the simulation agents to keep things working within the virtual village. Simulation agents can be simple in a limited environment, but as the complexity of the situation grows, the agents may have to be rewritten again and again to compensate for new factors. The amount of testing required to implement new conditions may become prohibitive at some point.

The second potential issue is the use of file systems and progress-saving. We will not be implementing a server-side storage system that the game itself would require in the real world, but we still will need to develop a way to store the state of a village locally. This is a process that neither member of the team has any experience with, so progress on a storage solution may be quite difficult. This will represent a situation of extending beyond our curriculum.

**Team Makeup**

* Matthew Callen
  + Leader, scribe, documentation, design, development
  + Game programming degree
  + Experience with Unity engine and language to be used, C#
  + Two self-started independent game design projects
  + Career goals in independent development
* Khangal Erdnetsogt
  + Version control, tracking, development
  + Computer science degree
  + Experience with Unity engine and language to be used, C#
  + Experience with tools such as Photoshop, Illustrator, 3DSMax
  + Past work as webmaster and software development