Technical Design Document

Pibblie Pums

Greg VanKampen

2/22/2019

Document History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author(s) | Changes |
| 0.1 | 2/21/2019 | Greg VanKampen | Initialization |
|  |  |  |  |
|  |  |  |  |

Table of Contents

Contents

[1. Game Summary 3](#_Toc1665086)

[2. Development Environment 3](#_Toc1665087)

[I. Development Hardware 3](#_Toc1665088)

[II. Programming Language 3](#_Toc1665089)

[III. Development Tools 3](#_Toc1665090)

[IV. External Code 3](#_Toc1665091)

[V. Game Engine 3](#_Toc1665092)

[3. Architectural Analysis 4](#_Toc1665093)

[I. Classes 4](#_Toc1665094)

[II. Behavioral Analysis 5](#_Toc1665095)

[III. Game Loop 7](#_Toc1665096)

[4. Technical Risks 7](#_Toc1665097)

# Game Summary

Virtual Pet very similar to a Tamagotchi. Raise your pet from baby to adulthood. The way you raise your Pibblie Pum will determine what it will be when it grows up. Feed it different foods. Play games with your pet. After some time, it will evolve into a totally new form. Try different pet raising methods to see all PibbliePums.

# Development Environment

## Development Hardware

PC- OS: Windows 10

## Programming Language

C++15

JSON

## Development Tools

Visual Studio 15.9

Aesprite 1.2

## External Code

JSON for Modern C++ 3.2.0

## Game Engine

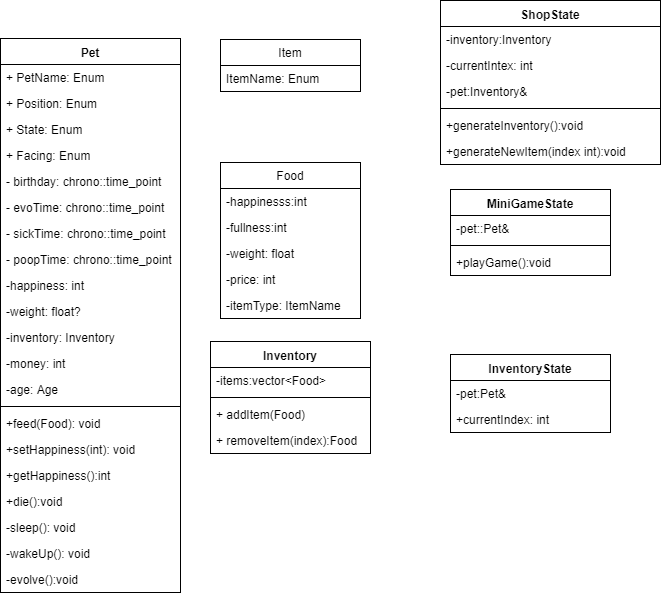
GEX Engine

# Architectural Analysis

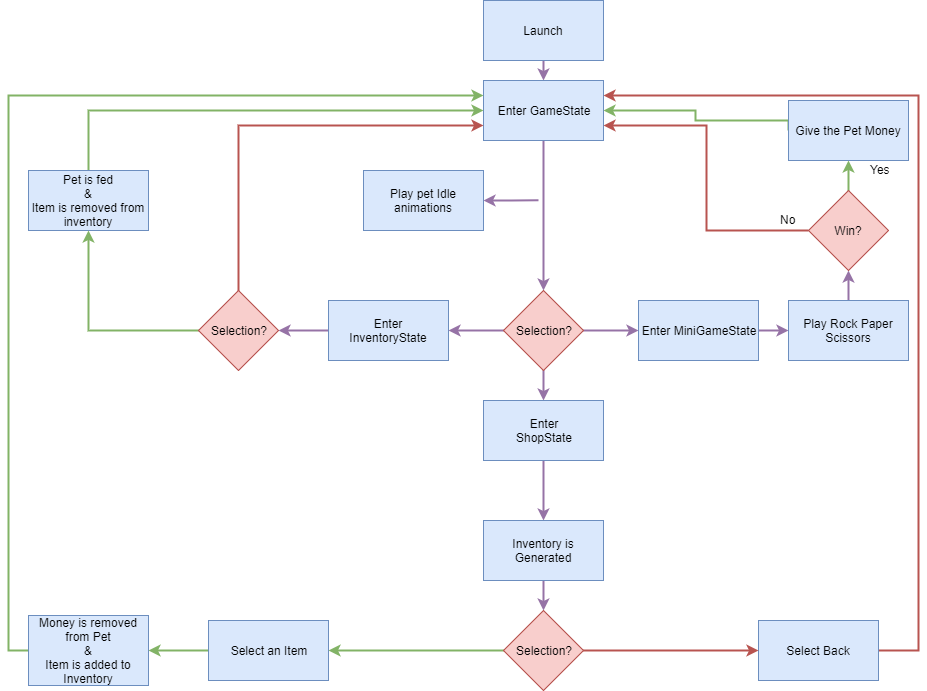
## Classes

|  |  |  |
| --- | --- | --- |
| Class | Responsibilities | Collaborations |
| Pet | Control the pet’s animations and change its state & stats | Inventory |
| Inventory | Manage and hold Food objects | Food |
| Food | Contains the stats changes a food item applies to a pet |  |
| ShopState | Displays, generate, and trade items to be sold to the player | Inventory |
| InventoryState | Display and use items in Pet’s inventory | Inventory |
| MinigameState | Play Minigames |  |

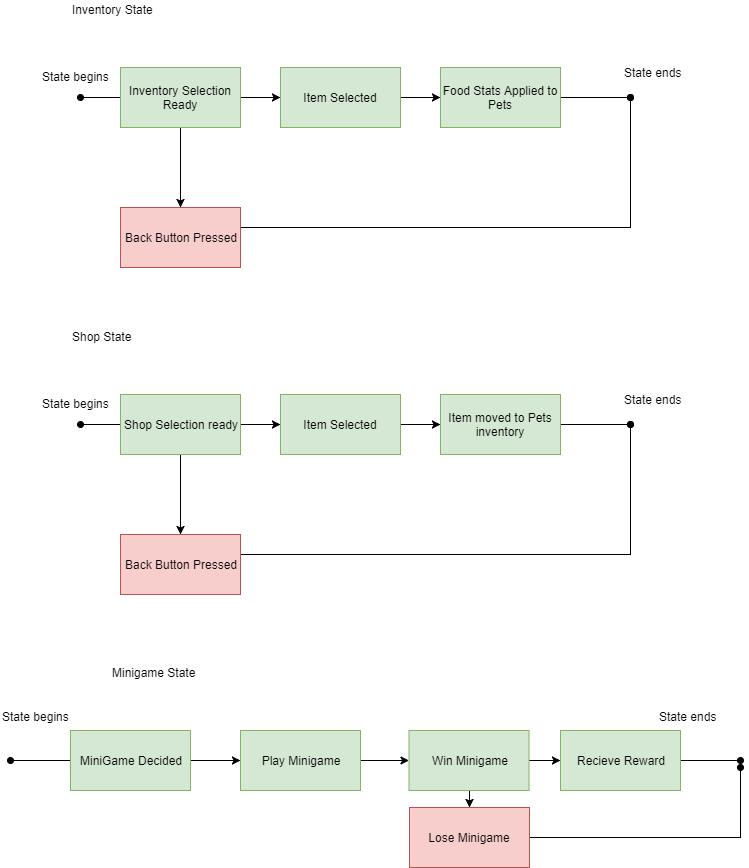
**UML**



## Behavioral Analysis

Activity diagram

Statechart



## Game Loop

* Loop Begins
* Push any new States to the StateStack
  + Pause GameState
  + Handle inputs for new state
  + Check for state pop request
* Handle Menu Navigation Inputs
* Update Pet
* Render screen
* Return to start of Loop

# Technical Risks

|  |  |  |
| --- | --- | --- |
| Risk | Severity | Mitigation (what is to be done to eliminate or minimize this risk) |
| First time doing sprite art | Low | Allotting extra time to improve |
| Uncertain how to implement pet timed events | Low/Medium | Research C++ time points |
| Uncertain how to implement Minigame loop on a new state | High | Follow the design pattern of the World and Gamestate class to try to replicate it |