

|  |
| --- |
| Stanford University  Administrative Systems  Functional Specification Document  Procedural Animation / Soft Body |

**DOCUMENT VERSION 1.1**

**6/25/2025**

**AUTHORS**

|  |  |  |
| --- | --- | --- |
| **Name** | **Role** | **Department** |
| Douglas Napier | Lead | My own |

**DOCUMENT HISTORY**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Document Revision Description** | **Document Author** |
| 06/25/2025 | 1.0 | Filled out document partially, 1.1 – 1.6 | Douglas Napier |
| 06/30/2025 | 1.1 | Filled out document partially, 2.1 – 2.2 | Douglas Napier |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**APPROVALS**

|  |  |  |  |
| --- | --- | --- | --- |
| **Approval Date** | **Approved Version** | **Approver Role** | **Approver** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

[1. Introduction 3](#_Toc345929872)

[1.1 Purpose of the document 4](#_Toc345929873)

[1.2 Project Scope 4](#_Toc345929874)

[1.3 Scope of the document 4](#_Toc345929875)

[1.4 Related documents 4](#_Toc345929876)

[1.5 Terms/Acronyms and Definitions 4](#_Toc345929877)

[1.6 Risks and Assumptions 4](#_Toc345929878)

[2. System/ Solution Overview 5](#_Toc345929879)

[2.1 Context Diagram/ Interface Diagram/ Data Flow Diagram, Application Screen Flow, Sitemap, Process Flow 5](#_Toc345929880)

[2.2 System Actors 5](#_Toc345929881)

[2.3 Dependencies and Change Impacts 5](#_Toc345929882)

[3. Functional Specifications 5](#_Toc345929883)

[3.1 <Title> 6](#_Toc345929884)

[4. System Configurations 8](#_Toc345929885)

[5. Other System Requirements/ Non-Functional Requirements 9](#_Toc345929886)

[6. Reporting Requirements 9](#_Toc345929887)

[7. Integration Requirements 9](#_Toc345929888)

[7.1 Exception Handling/ Error Reporting 9](#_Toc345929889)

[8. Data Migration/ Conversion Requirements 10](#_Toc345929890)

[8.1 Data Conversion Strategy 10](#_Toc345929891)

[8.2 Data Conversion Preparation 10](#_Toc345929892)

[8.3 Data Conversion Specifications 10](#_Toc345929893)

[9. References 10](#_Toc345929894)

[10. Open Issues 10](#_Toc345929895)

[Appendix 10](#_Toc345929896)

# Introduction

*Rain world has a very interesting ecosystem; the problem is to work on recreating procedural animation like what Rain World uses. Procedural animation involved math and physics simulations. The problem will be to create a piece of software using the python programming language that creates a window for the user and enables the use of procedural animation and soft body physics.*

## Purpose of the document

The Functional Specification Document is a document that provides detailed information on *how* the system solution will function and the requested behavior. This document is created based on the high-level requirements identified in the Business Requirements Document and provides traceability on the functional specifications back to the business requirements. Included in this document will be the detailed functional requirements including use cases, system inputs and outputs, process flows, diagrams, and mock ups.

## Project Scope

*Soft body simulation:*

*Model a 2d soft body as a grid of particles (ie: 5x5) connected by distance constraints.*

*Use verlet integration for stable physics updates of particle positions*

*Implement basic physics:*

*Gravity (Constant downward force in pixels per second squared)*

*Ground collisions (Particles bounce or rest at some value being the window’s bottom)*

*Distance constraints to maintain the soft body’s structure while allowing deformation*

*Support a small scale simulation (limited amount of particles) for performance.*

*Procedural Animation:*

*Add lifelike motion to the soft body, inspired by Rain World’s organic animations.*

*Implement simple procedural animation via periodic forces. (such as forces applied to specific particles)*

*Allow for future extensibility.*

*Custom Display Systems:*

*Render the simulation using Python’s tkinter standard library without external libraries like pygame.*

*Draw particles as small circles and constraints as lines*

*Create a window with a black background, white lines, and white circles for visibility.*

*Achieve smoother rendering.*

## 1.5 Terms/Acronyms and Definitions

|  |  |  |
| --- | --- | --- |
| **Term/Acronym** | **Definition** | **Description** |
| Soft Body | A deformable object in a simulation that can bend, stretch, or squish, unlike a rigid body such as a cube that maintains a fixed shape. | The soft body will be modeled as a grid of particles connected by virtual springs or constraints. Think of a jelly-like blob that wiggles when poked or falls under gravity. |
| Particle | A point in 2d space with a position (x, y coordinates), a previous position (for tracking motion), and a mass. Particles will be the building blocks of the soft body. | Each particle is a tiny dot in a grid, moving independently but tied to the other particles. The particles form the mesh that defines the body’s shape and movement. |
| Verlet Integration | A numerical method for updating the positions of particles in a physics simulation. It uses a particle’s current and previous positions to calculate its next position, included forces like gravity. | Is stable for simulations with constraints. For instance, if a particle falls, verlet integration will predict where it will land based on where it was and how fast it’s moving. This helps animate the soft body’s particles over time. |
| Distance Constraint | A rule that keeps two particles at a specific distance by adjusting their positions. | It mimics a spring or rubber band, so if two particles drift far away, constraint pulls them closer. If too close, pushes them apart. The constraints maintain the body’s structure while allowing some deformation. |
| Rest Length | The ideal distance between two particles connected by a distance constraint, measured when the soft body is at rest. | If two particles are 50 pixels apart, that’s their rest length, and the constraint tries keeping them at that distance. |
| Procedural Animation | Animation generated algorithmically rather than through pre-designed or hand-animated animation. | In this project, it involves applying forces or movements to particles to create lifelike motion. This adds dynamic, natural movement to the soft body. |
| Sinusoidal force | A force that varies over time according to a sine wave, creating smooth motion. | Applying a force that alternates left and right to make a soft body wiggle, it’s used in procedural animation to create rhythmic, lifelike movements. |
| Time Step (Δt or dt) | The small interval of time between each update of the simulation. The triangle symbol is known as delta, referencing change. So delta t refers to the change in t, t commonly being time in seconds. | This determines the frequency the physics calculations are performed. So a time step of 1/60 seconds means the simulation updates 60 times a second. This controls the smoothness and stability of the simulation. |
| Constraint iterations | The number of times the simulation adjusts particle positions to satisfy distance constraints each frame. More iterations improves accuracy but increases computation. | Furthermore, more iterations per frame adds more stiffness and rigidity to those constraints. This enhances the soft body’s stability and stiffness. |
| Tkinter | This is python’s standard library for creating graphical user interfaces. In this project we rely on the Canvas widget to draw the simulation. | Tkinter creates a window where the soft body is visualized, it’s the tool for rendering the simulation without external libraries like pygame. |
| Canvas | A tkinter widget that provides a drawing area for rendering shapes like lines, ovals, and polygons. | The surface for display the simulation. |
| Frames per second | The number of times the simulation is updated and redrawn per second, measured in frames per second (FPS). | This determines how fluid the animation appears. More is better, depends on monitor refresh rate to notice the difference. |
| Fixed particle | A particle with infinite mass that doesn’t move, used to anchor the soft body. | This prevents the soft body from falling entirely under gravity, it is like the ‘core’ of the soft body. |
| Ground Collision | A rule that prevents particles from moving below a defined ground level. Particles are repositioned to rest on or bounce off the ground. | If a particle falls to the bottom of the window it will stop instead of disappearing, acting as a floor. |

## 1.6 Risks and Assumptions

*Risks are issues that can disrupt the project’s implementation. For instance:*

*Performance limitations with tkinter. Tkinter is python’s standard GUI library, it isn’t optimized for real-time graphics or animations. Rendering a soft body with many particles or frequent redraws may lead to lag, stuttering, or low frame rates. We can mitigated this by using a small grid, optimizing rendering, capping the time step, or try using numpy for physics.*

*Simulation instability. The verlet integration and constraint-based physics can become unstable under certain conditions, such as jittering, weird stretching, or exploding. This would undermine realism and procedural animation. We can mitigate this by capping dt at 0.02 seconds to ensure smaller updates, or use 10 or more constraint iterations a frame to enforce rest lengths. We can also use damping (Reducing velocity slightly each frame) to prevent oscillations from growing.)*

*Limited debugging tools for physics. Without a game engine or physics library, debugging issues like constraint violations or particle misplacement requires manual logging or visual inspection. We could mitigated by adding logging, such as printing particle positions or errors.*

*Assumptions are conditions taken for granted in the design, for instance:*

*We assume tkinter will be able to handle real-time rendering. If we’re wrong, there may be low frame rates or laggy visuals that require reducing the simulation’s complexity, or we may need to use external libraries.*

*We assume simple procedural animation is sufficient. We may have really mechanical or simplistic animation. If so, we may need more advanced techniques like inverse kinematics or noise functions.*

*We have a big constraint, primarily being:*

*No external libraries. We don’t (or I don’t) want to use external libraries such as Pygame, or OpenGL. I want to only use standard libraries packaged with Python itself. This may lead to worse performance than if we used an optimized library. >*

# System/ Solution Overview

*The software is a 2d soft body simulation built in Python, utilizing the Tkinter library for rendering. A soft body, modeled as a grid of particles connected by constraints, deforms realistically under forces like gravity. Procedural animation generates lifelike, organic movements algorithmically, and the simulation will be visualized in real time within a Tkinter window.*

*The solution aims to provide an educational and visually engaging tool to demonstrate soft body physics and procedural animation principles. It enables users to explore deformable object behavior, making it ideal for integration into larger projects.*

*The objectives and goals are to:*

*create a stable simulation using Verlet integration and distance constraints.*

*To implement procedural animation for natural, responsive soft body motion.*

*To render efficiently at ~60 FPS in Tkinter’s canvas.*

*To ensure the code is modular and clear for future extensions.*

*To produce a functional prototype showcasing soft body dynamics, balance realism for performance, and to establish a flexible foundation for extensions, like user controls or inputs.*

## 2.1 System Actors

### 2.1.1 User Roles and Responsibilities / Authority Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **User/Role** | **Example** | **Frequency of Use** | **Security/Access, Features Used** | **Additional Notes** |
| *User of the software* | Douglas Napier wants to simulate physics, so he will use the software. | *The system will be used rarely.* | *No special security needed* |  |

## 2.2 Dependencies and Change Impacts

### 2.2.1 System Dependencies

*<List and identify any dependencies the proposed solution will have on other systems.>*

*The dependencies are as follows:*

*Python 3. The simulation is written in python, so an up-to-date python installation is required on the user’s system. This is the core dependency since the code won’t run without it.*

*Tkinter library. Tkinter is required to render the graphical user interface (GUI) for simulation. It’s included in most Python installations, but on some systems, it may require installation.*

*Hardware. The simulation requires basic hardware, a CPU, RAM, and a GPU.*

*Packaged .exe. This will also be offered as a .exe file via PyInstaller, a .exe would only be executable on Windows systems. The .exe will bundle Python, Tkinter, and necessary libraries, this would provide a simplified deployment but restrict the software to Windows environments.*

# Functional Specifications

*<Start describing the specifications related to the overall system here. You may want to create a table/ index of all functionalities explained in the sections below and link them to the items below>*

*<If no separate reference/ traceability document is created for the project, use this section to map the business requirements, use cases, functional requirements and the test cases>*

*<Group your functional specifications as appropriate for your project. You may want to divide them by screens, functional areas, user role, JIRA tickets or high-level functions Vs detailed functions or any other way that works for your project>*

## 3.1 <Title>

### 3.1.1 Purpose/ Description

*<Include a high-level description and purpose of the specifications covered in the section.>*

### 3.1.2 Use case

*<Map the functional requirement to one or more use cases mentioned in the Business Requirements document. If the use case is not described in detail in the Business Requirements document, describe the use case here. This typically includes the element s in the following table.>*

|  |  |
| --- | --- |
| **UC-1** | **<Use case name>** |
| **Primary Actor(s)** | *< primary actors that participate in this use case>* |
| **Stakeholders and Interest** | *<One sentence describing other stakeholders>* |
| **Trigger** | *<Condition/action that initiates/starts the use-case>* |
| **Pre-conditions** | *<Condition assumed to be true before the first step>* |
| **Post-conditions** | *<Condition after the use case is successfully executed >* |
| **Main Success Scenario** | 1. *<visit STARTING-POINT* 2. *Step* 3. *Step* 4. *Make sure GOAL-ACHIEVED>* |
| **Extensions** | If Condition, then Alternative Steps  <List any extended steps/ scenarios that occur, other than the main success scenario.> |
| **Priority** | *<indicate priority of high, medium or low)* |
| **Special Requirements** | *<Any system related special requirements needed to fulfill the use case>* |
| **Open Questions** | *<Notes and questions>* |

### 3.1.3 Mock-up

*<Provide the mock-up of the functionality or a mock-up of the entire page>*

### 3.1.4 Functional Requirements

*<Describe the page level details that are not captured in section 3.1.5 below. These may include any requirements related to Navigation Menu, Actions, transaction status, verification and validation requirements etc.*

*Make sure each specification has a reference number and is explained in the following format.>*

|  |  |  |
| --- | --- | --- |
| **Spec ID** | **Specification Description** | **Business Rules/ Data Dependency** |
| *<Specification Identifier>* | *<Short explanation of the specification>* | *<Any validation rules or business rules>* |

*<Note: Section 3.1.4 and section 3.1.5 may be combined if there are a few functionalities on a particular page>*

### 3.1.5 Field level specifications

*<Specify all the field data elements related to the functional requirement in both tables below.>*

**Form Elements:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Call-out** | **Field Label** | **UI Control** | **Mand?** | **Editable** | **Data Type** | **Value Set** | **Default Value** | **Data Example** | **Data Source** |
| *<mock-up reference>* | *<Label name>* | *<specify what UI control will be on screen>* | *<specify if field is mandatory>* | *<specify if field is editable>* | *<Specify the data type that will be used for this field>* | *<If value is from the set, specify the entire value set here>* | *<Specify if it should be defaulted to any value>* | *<Provide an example of the data>* | *<Specify the source of the data>* |
| Example:  Call-out 1 | User name | textbox | Yes | Yes | Alpha-numeric | none | NA | agujar | User entry |

**Form Business Rules and Dependencies:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field Label** | **Validation / Business Rules** | **Error Messages** | **Data Dependencies** | **Additional Info/ Notes** |
| *<Label name>* | *<Specify the validation rules and/ or business rules applicable to the form element>* | *<List the error message that should be displayed and under what conditions>* | *<Specify if there are any date dependencies>* | *<Provide any additional information here>* |
| *Example: User name* | *User name shall be a valid Stanford Sunet ID* | *For incorrect user name display on setFocusOff: “Please provide a valid user name”* | *None* | *Access prohibited only to Stanford affiliates. For non-Stanford affiliates, check call-out 3.1* |

**Buttons, Links and Icons:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Button, Link, Icon Label** | **OnClick Event** | **Other Event** | **Visible** | **Enabled Vs Disabled** | **Navigate To** | **Validation** | **Dependencies** |
| *<Button label name>* | *<Specify the operation that would be performed on an on-click event>* | *<Specify the operation that would be performed on other events>* | *<Specify default visibility of the button>* | *<Specify if button is enabled or disabled and the condition, if any>* | *<Specify the link where the page will be re-directed, if any>* | *<Specify the validation rules on operation of the button>* | *<Mention if there are any dependencies on other form elements and/ or buttons>* |
| *Example: Submit* | *Verify if user name and password are correct. If yes, log the user into the system.* | *OnMouseHover display the following message: “Please provide your web-authentication details to sign in>* | *Yes, always* | *Disabled, by default.*  *Enabled, after first key entered in either username or password field.* | *User Dashboard page* | *Verify if Username is a valid sunet ID and user name and password match with registry data.* | *Disable the New user functionality on subsequent pages if user logs in through this button.* |

# System Configurations

*<Provide an overview of all the steps or the set ups required to configure an application/program. Also state the intent or purpose behind each set up or configuration. Discuss the possible alternatives, customizations, workaround’s, conditions and dependencies in a particular configuration. In case of Oracle applications, please list all applicable BR100’s or Application set up documents>*

# Other System Requirements/ Non-Functional Requirements

*<This section is used in contrast with stated functional requirements to highlight the additional details on the quality related aspects as well as other behavioral aspects of a system. This section is used to capture the stakeholders’ implicit expectations about how well the system will work under a given circumstance. Here you can state the specific SLA’s related to system response times (Data search and retrieval), Performance needs and metrics, Latencies in a particular timeframe or during high volume transactions, System failures and recovery management, Security levels and accessibility constraints, Data Backup and archiving Capabilities, Legal compliance needs etc. The broader definition of the term ‘system’ also includes integrations with all types of Mobile platforms, Mobile devices, Tablets and Smart phones.>*

# Reporting Requirements

*<This section is used to capture the reporting needs, including but not limited to the scope and format of the report, data elements and contents required on the report, file types and extraction mechanisms, user base and accessibility levels, frequency of report extractions etc. Also provide the mock up of the report if needed. If necessary, create a separate document for reporting requirements.>*

# Integration Requirements

*<Identify the integration needs and state all required interfaces with anything external to this solution including hardware, software, and users. Include Architectural overview diagrams, high level data flow diagrams, table structures and schema, interface protocols, API’s, Error conditions, Error validations and messaging needs, Auto processing requirements etc. You can optionally state hardware and software dependencies, Upgrade requirements, compatibility issues with existing frameworks and solutions, etc>*

*(Data Flow Diagrams, Interface Diagrams – if necessary)*

## Exception Handling/ Error Reporting

*<This is where you can explain the error conditions/Exceptions that normally happen in Interfaces or cross flow system integrations. Explain the nature of exception, Error Id, Root cause of the error and also the strategy to handle the scenario. You can also indicate if there are any concurrent programs designed to automatically handle the error records or error conditions. State if there are any error reports generated or notifications utilized to alarm the support teams and system Administrators during the interface failures or outages>*

|  |  |  |  |
| --- | --- | --- | --- |
| **Exception/ Error ID** | **Error** | **Cause** | **Solution Strategy** |
|  |  |  |  |

# Data Migration/ Conversion Requirements

*<Explain in brief the data conversion plan. Provide full identifying information for the automated system, application, or situation for which the Data Conversion Plan applies. Describe briefly any assumptions, constraints or risks regarding the data conversion effort. (Provide details in section 1.6)>*

## Data Conversion Strategy

*<Include the overall strategy for the Data Conversion. This includes how and when you will perform the conversion - the approach used to extract, transform and load data during the conversion process, the conversion schedule, and test plan for testing the converted data.>*

## Data Conversion Preparation

*<Provide details on any prerequisites necessary for the conversion. Discuss the backup strategy, restoration process in case the conversion fails.>*

## Data Conversion Specifications

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Source** | **Source Data Element** | **Target** | **Target Data Element** | **Conversion Rules** | **Notes** |
| *<Source location>* | *<Source Data Element Identifier >* | *<Target location>* | *<Target Data Element Identifier>* | *<Describe rules for Data conversion>* | *<Additional notes>* |

# References

*<List all references to external material used as background information or knowledge for the FSD. Examples may include a compliancy website, Stanford website, etc>*

# Open Issues

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Issue ID** | **Issue** | **Raised By** | **Raised On** | **Solution/ Decision** | **Resolved By** | **Resolved On** | **Status** |
|  |  |  |  |  |  |  |  |

# Appendix