https://images-na.ssl-images-amazon.com/images/I/71ddDTpSkSL._SL1134_.jpg

# DinoTracks
## Thrive To Survive

By Dan Phillips, Cindy Li, Ben Krupka, and Kevin Yuan

9/26/19 Revision 1
10/22/19 Revision 2

# Executive Summary

The game is based on islands during the Mesozoic period. Being limited on time while also trying to manage resources and fight other dinosaurs is what makes the game interesting. This game should be published because it combines the awesomeness of dinosaurs with classic survival and roguelike mechanics.

# Overview

**Title**: DinoTracks
**Tagline**: Thrive To Survive
**Names**:

- Dan Phillips
- Ben Krupka
- Cindy Li
- Kevin Yuan

**Genre**: roguelike.
**Platform**: PC
**Market**:

- Age: 18-22.
- Gender: all.
- Interests: dinosaurs, survival, strategy.
- Occupation: students.

**Setting**:

- Where: a remote island (climate varies based on player selection).
- When: shortly before the meteor that ends the Dinosaurs' reign strikes.
- Conditions: pre-apocalyptic (player's dinosaur knows).
- Tone: comic, lighthearted.

**Plays Like**: NetHack meets dinosaurs.
**Summary**: players control a dinosaur that has somehow gained knowledge that a meteor will soon strike and wipe them out. They must explore, eat, and possibly fight their way to a place that will somehow protect them from the impact. This is complicated by the time limit of the asteroid, along with players having to maintain their dinosaurs' energy level through eating and avoid predators that want to eat their dinosaur.
**Mechanics**:

- Selecting an island and dinosaur.
- Move and explore.
    - Collecting powerups.
- Health, energy, and time management.
- Combat (both offense and defense).
- Other dinosaurs also perform actions.

**Reference Art**:

- Jurassic Park/World (books/movies).
- Terraria.
- Nethack.
- Dwarf Fortress.
- Crypt of the Necrodancer.
- Rimworld

**Related Games**:

The Isle, Afterthought LLC, Simulation/PC, 2015:

- Link to Steam: https://store.steampowered.com/app/376210/The_Isle/
- Interesting point: This game's idea is similar to ours. This game is about controlling a dinosaur to survive and evolve on an island.

Don't Starve, Klei Entertainment, Survival/PC, 2013

- Also a top-down survival game. Although Don't Starve focuses more on crafting and building to aid survival, both games are about managing your time and energy to survive as long as possible.

Wayward, Unlok, Roguelike/PC, 2016

- Both games focus on survival with the players actions affecting their stats and chance at survival. Wayward has no levels and players roam the map freely as they collect items, whereas our game will have a time limit and rounds.

# Related Games

## The Isle

Afterthought LLC, 2015

This game is similar to ours in that both are about dinosaurs. More specifically, both involve the player being a dinosaur that has to survive on an island.

This game is different from ours in that it is multiplayer, with PvP combat between the dinosaurs of other players, while our game's combat will be versus AI since it is singleplayer. This game also has more sandbox elements than ours, which will focus on survival. Related to both of these is that The Isle is far more open-ended than our game, which has the time limit imposed by the meteor. Another difference is the fact that this game is 3D while ours is 2D.

Another similarity is that in both games, dinosaurs are able to consume other dinosaurs (or, in our case, the "meat item" dropped by dinosaurs) in order to regain energy.

While the goal of both games is to survive, DinoTracks has an end goal and a time limit in order to win the game; The Isle is just that, survive. In The Isle, the goal is to strengthen the player's dinosaur and survive among the other players.

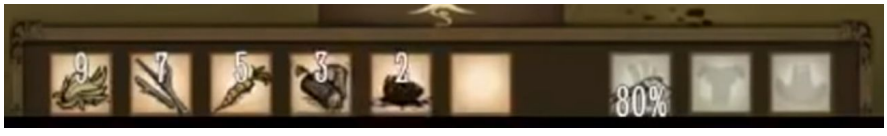
## Don’t Starve

Klei Entertainment, 2013

This game is similar to ours in that it involves managing resources like health and hunger. It is also roughly top-down. Both games also involve combat that is in-world without moving to a separate combat screen and system. Also, both games have predefined characters (although ours are dinosaur genera/species rather than humans/robots/monsters).


This game is different from ours in that it is more open-ended with the main goal being to survive as long as possible. Don’t Starve also has crafting and building mechanics that our game won’t have. Also, Don’t Starve’s maps are procedurally-generated while our maps will be hand-made with locations of items and enemies randomized.


Similar to DinoTracks system of having an inventory of powerups; however, in Don’t Starve, the player has an inventory of supplies to use in crafting and building

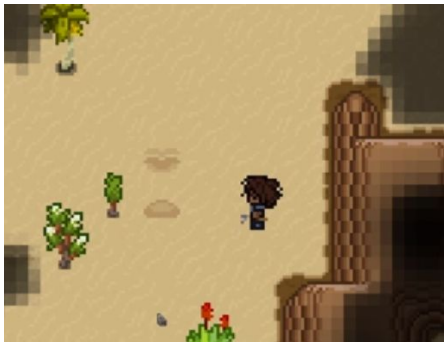
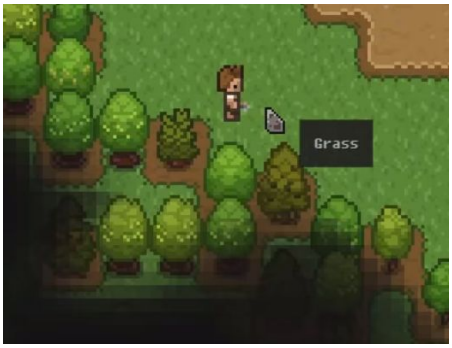


## Wayward

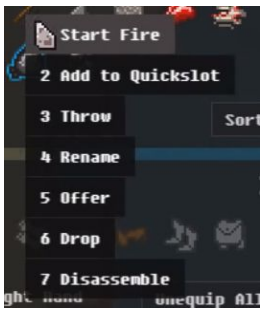
Unlok, 2016

Similar to DinoTracks, Wayward is a top-down survival game where the player collects resources that will affect their stats. Both games are turn-based, where the actions in the world around the player only happen when the player performs an action. Another similarity is that both games have in-world combat, with the combat being triggered as the player approaches the enemy. In both games, if the player defeats the enemy, they will receive items that can be used. If the player's health runs out during combat, the game ends.


Like DinoTracks, Wayward has different environment types to explore. The terrain for both have obstacles and vegetation. In addition to that, both have/will have terrain types that affect the player's movement around the map.


One of the biggest differences is that while DinoTracks has a time limit and an end goal in the form of the meteor strike, Wayward is open-ended and does not have a goal other than character development and survival. Also, Wayward has crafting and building elements, as well as skills that DinoTracks will not have. The map for DinoTracks will also not be procedurally generated, as it is for Wayward.



## Crypt of the Necrodancer

Brace Yourself Games, 2015

Both games are top-down roguelike games with combat and item collection that focus on the player moving in a turn-based fashion. Both games also have a health system to indicate how much more damage the player can take.

Unlike DinoTracks, the progression of Crypt of the Necrodancer is a series of levels where the player collects weapons, armor, and treasure.


While Crypt of the Necrodancer requires the player to move with a rhythm and has the other aspects of the game happen according the rhythm, our game will have it so that things only happen when the player moves or performs another action.

# Player Composition

- Victor Bennett: male, 16 years old, highschool student, love Jurassic Park movies, play pixel games for 1 hour everyday.
- Philip Williams: male, single, 28 years old, doing research in dinosaurs with a professor in archaeology, play video games for 3 hours a week.
- Sharon Anderson: female, single, 22 years old, love survival games, especially don't starve, play video games after work for 2 hours a day.
- Alice Howard: female, single, 26 years old, huge fan of dinosaurs, buy lots of dinosaur models, go to the dinosaur section in every natural museum.

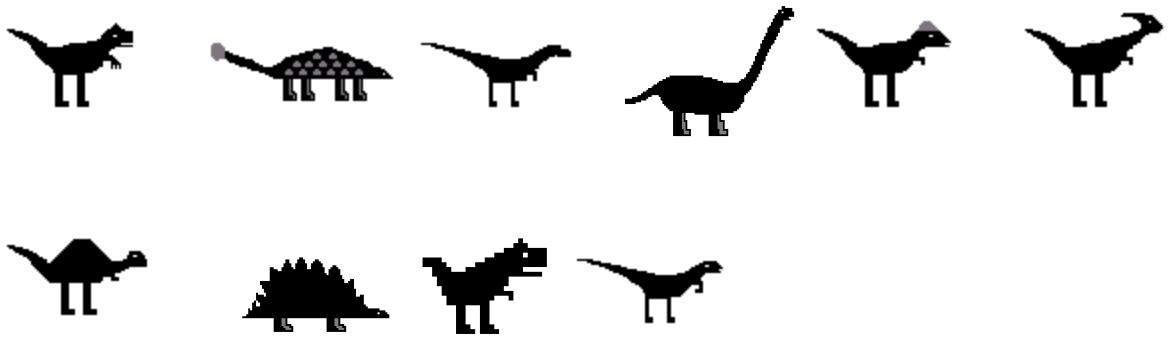
# World

- Islands during Mesozoic period filled with plant life and dinosaurs.
    - Dinosaurs are the only animals on the islands.
- Different climates of islands affect vegetation and other dinosaurs that are present.
- A meteor is going to collide with the Earth and wipe out many groups of life.
- The player's dinosaur knows about the dinosaur because it has been granted advanced intelligence by aliens that are trying to preserve life from disasters.
- The aliens have provided an escape pod that will bring the player's dinosaur up to their ship, and the player needs to survive and traverse the island to get there before the meteor hits.
- The aliens have also provided a locator that beeps faster when the player's dinosaur is closer to the escape pod.
- The aliens will do a final determination of the player dinosaur's worthiness when it reaches the escape pod.

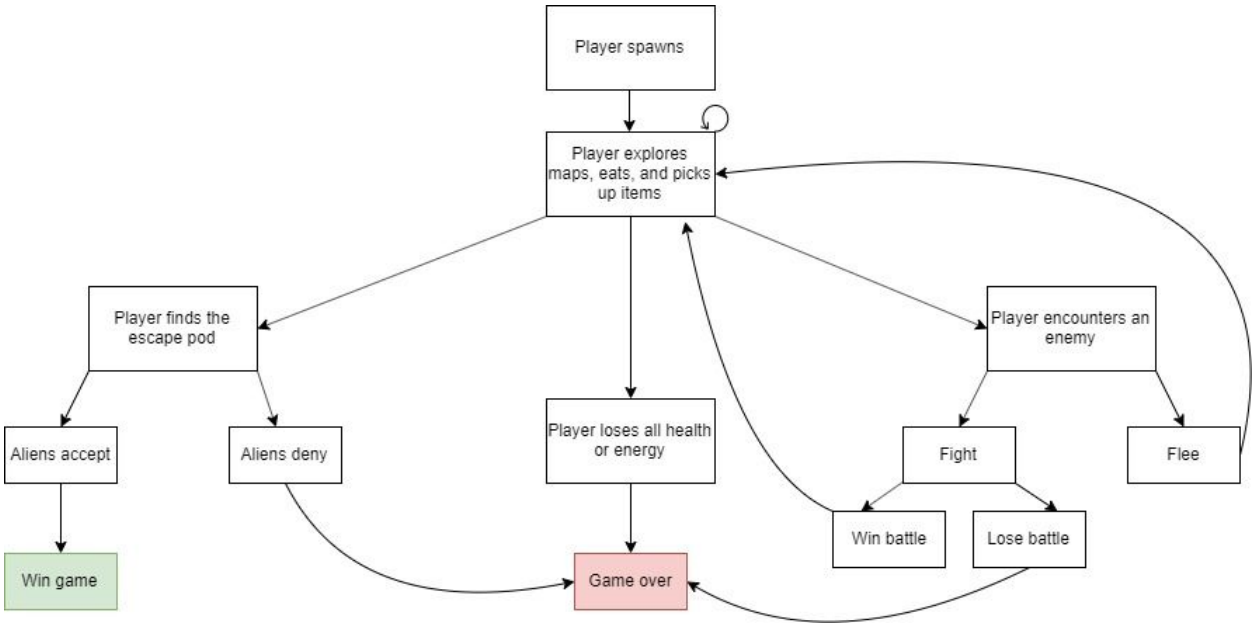
# Characters

- Carnivores:
  - Tyrannosaurus
    - Large.
    - Medium speed.
    - High damage.
    - Low defense.
  - Velociraptor
    - Small.
    - High speed.
    - Medium damage.
    - Low defense.
  - Carnotaurus
    - Medium.
    - Medium speed.
    - Medium damage.
    - Low defense.
  - Spinosaurus
    - Large.
    - Slow speed.
    - High damage.
    - Low defense.
  - Allosaurus
    - Medium.
    - High speed.
    - Low damage.
    - Low defense.
  - Coelophysis
    - Small.
    - Medium speed.
    - Low damage.
    - Low defense.
- Herbivores:
  - Ankylosaurus
    - Large.
    - Slow speed.
    - Medium attack.
    - Very high defense.
  - Stegosaurus
    - Large.

        - Slow speed.
        - Medium attack.
        - High defense.
    - Pachycephalosaurus
        - Medium.
        - Medium speed.
        - High attack.
        - Medium defense.
    - Protoceratops
        - Small.
        - High speed.
        - Low attack.
        - Medium defense.
    - Parasaurolophus
        - Medium.
        - Very high speed.
        - Low attack.
        - Low defense.
    - Diplodocus
        - Very large.
        - Very slow speed.
        - Low attack.
        - Medium defense (and a lot of health)



# Plot Graph

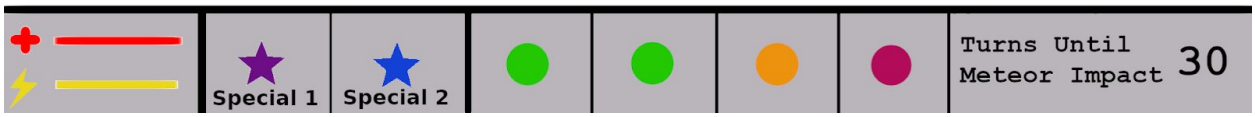

Player spawns
Player explores maps, eats, and picks up items
Player finds the escape pod
Player encounters an enemy
Aliens accept
Aliens deny
Player loses all health or energy
Fight
Flee
Win battle
Lose battle
Win game
Game over

# Art Direction

- Pixel art for characters and everything else in-world.
- Background tiles maybe more detailed?
- Top-down.
- Escape pod should be very visually distinct.
- Radar that beeps faster the closer one is to escape pod.

# UI Storyboards

## Current Map View



# Tags and Dialogs

Dialogs:

- None so far.

Tags:

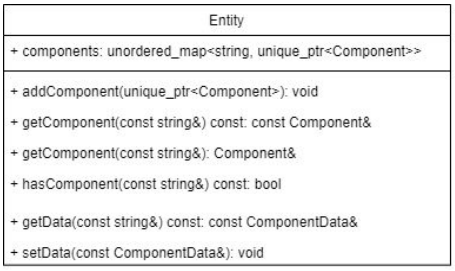
- Textures:
- Fonts:
- Actions:
    - MOVE_UP
    - MOVE_DOWN
    - MOVE_LEFT
    - MOVE_RIGHT

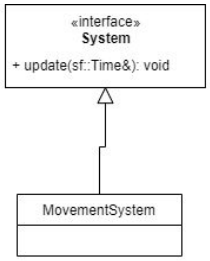
# Software Architecture

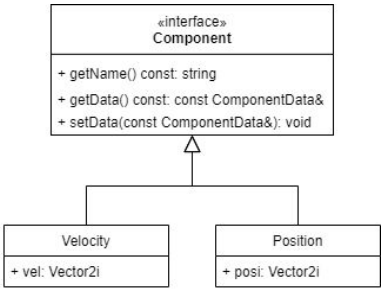
- Application:
    - ResourceManager: load fonts, textures, sounds, and other things from filesystem to provide to part of logic and views that need them.
        - Resources loaded at beginning.
        - Accessed globally (but not a singleton).
        - String keys identify resources.
        - Dinosaur types are also a resource since they are loaded from files. This means the files don't have to be read each time a dinosaur is created.
    - Variable timestep.
- Logic:
    - Islands:
        - Terrain: movement modifier, type (water, passable, vegetation, mountain)
            - Instances, not subclasses, represent a type of terrain.
        - Map is a grid of terrains.
    - Entity:
        - Components: position, velocity, health, damage, defense, energy, etc.
            - Each component has a string that identifies it.
            - Each component also has data associated with it that can be retrieved and set by systems.
                - Data will be a union for homogeneity between different types of components.
        - Entities contain maps of strings to components.
            - There are methods to add, get, and delete components from entities.
            - Entities can update/get data on components directly.
        - Enemies are same dinosaur types (will decide during later balancing whether to weaken them) as player.
        - Visual Component: stores string that the HumanView will use to decide appropriate texture for an entity.
    - Systems:
        - MovementSystem: moves entities (and calculates energy/health usage), checks collisions, determines if things are picked up, determines if combat happens.
        - HealthSystem: check for death, heal player, update health.
        - EnergySystem: check energy amount, update energy amount.
        - FoodSystem: restore energy.
        - CombatSystem: calculates damage, fleeing, energy usage.
    - EntityBuilder: makes creating components on Entities easier.
        - Create an instance and pass it an entity.

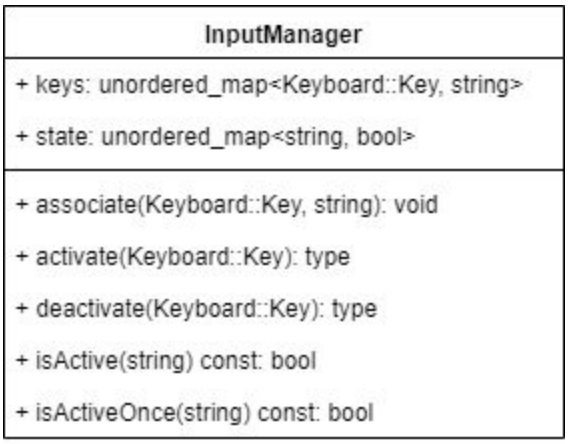
        - Call addComponent() methods with data to add components to entity.
        - Avoids manual handling of unique_ptrs.
    - Update:
        - Go through entities and run appropriate systems.
    - Turn timer for meteor.
    - Collision Detection:
        - Based on moving entities.
        - Checks against terrain and other entities.
        - Checks all spaces ahead in movement range.
        - We will check collision for moving entities against other moving entities and static entities.
        - If AIs collide with each other, the first one gets the space and the second is stopped as if it hit an obstacle.
        - Collision system will have to know what two types of things are colliding.
    - Dinosaur Types:
        - Vary based on speed, attack, defense, and starting health.
        - Load from a file.
    - Entity Ranges: how the entity vector in Logic is defined.
        - 0: player.
        - 1: escape pod.
        - 2-9: enemy dinosaurs.
        - 10-19: food.
            - 8 plant-based.
            - 2 for meat drops from enemy dinosaurs.
        - 20-24: powerups.
- View:
    - HumanView.
        - InputManager: maps player inputs to game actions and allows this mapping to be changed for configurability. Maps both keys and user clicks in the UI.
            - Actions currently represented by strings.
            - Map actions to keys and to bools.
            - When a key is pressed, the appropriate bool is set to true so that the appropriate command can be sent to
        - State machine for menus.
            - Title, Playing, GameOver.
        - Scrolling:
            - Use sf::View class.
    - AIView for enemies.
- Class Hierarchy:
    - Components all inherit from Component.
    - Entities created through composition of components rather than inheritance.
    - Possible shared base class for HumanView and AIView for common behaviors and shared functionality, such as receiving events from the EventManager.

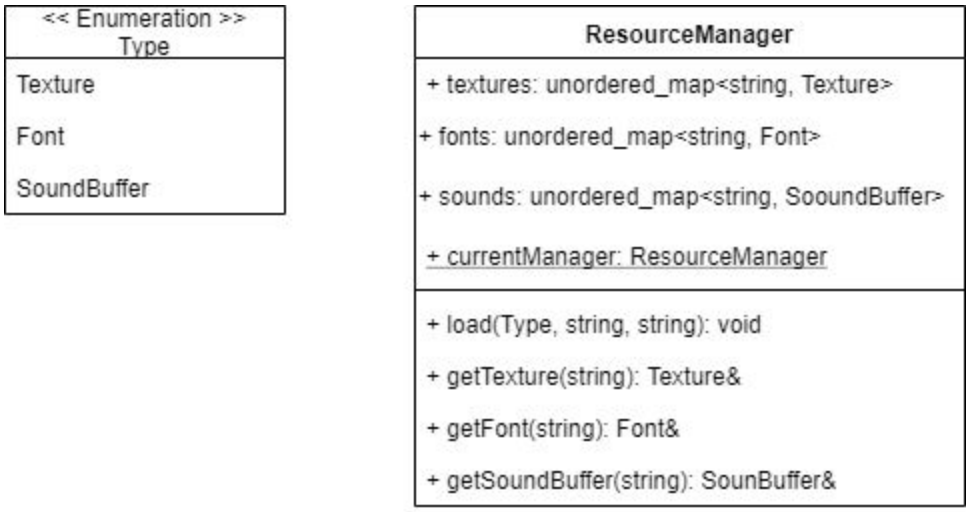
- Data Structures:
    - Maps: the maps is a 1D vector that is accessed as 2D. The map represents the terrain. Vegetation is considered as its own type of terrain for movement speed but is displayed on top of the base terrain for that map. obstacles Since the map sizes are constant throughout play, the map can have space reserved at loading time. The systems that access the map will use data from the map for things such as determining passability and movement cost.
    - Entity Locations: since entities (player, enemies, food, powerups) are sparse, their locations are a Component that will be accessed by systems that deal with location and movement. This means that they entity locations are stored separately from the map. The player will be entity zero.
- File Formats:
    - Map Names: x.map (e.g. 1.map, 2.map, etc…). Sored in resources/maps.
    - Maps: a 2D grid of space-separated integers. Each integer represents one tile of the map, with the integer value corresponding to a terrain type. This allows us to easily create maps with a text editor while also being fairly simple to load.
        - Ex: 0 0 0 0<br>0 1 2 0<br>0 1 1 0<br>0 0 0 0
        - Vegetation/obstacles and entities are generated and thus don't need to be part of format.
    - Key Configuration: a list of key-value pairs where the key is a game action and the value is the key. Each pair is on its own line and are separated by an '='.
    - Dinosaurs: name followed by health, speed, attack, defense (separated by spaces).
        - E.g: Stegosaurus 100 20 30 70
- Event Flow: turn-based.
    - One turn: user inputs action, their dinosaur does that action, then all other objects in the world respond to what they player did.
    - Meteor timer ticks down for every turn after everything in the world has performed their actions.
- External Libraries:
    - Thor: for actions, vector math, and maybe more.
- Diagrams (for visualizing more complex parts of architecture):


Entity
+ components: unordered_map<string, unique_ptr<Component>>
+ addComponent(unique_ptr<Component>): void
+ getComponent(const string&) const: const Component&
+ getComponent(const string&): Component&
+ hasComponent(const string&) const: bool
+ getData(const string&) const: const ComponentData&
+ setData(const ComponentData&): void


«interface»
System
+ update(sf::Time&): void
MovementSystem

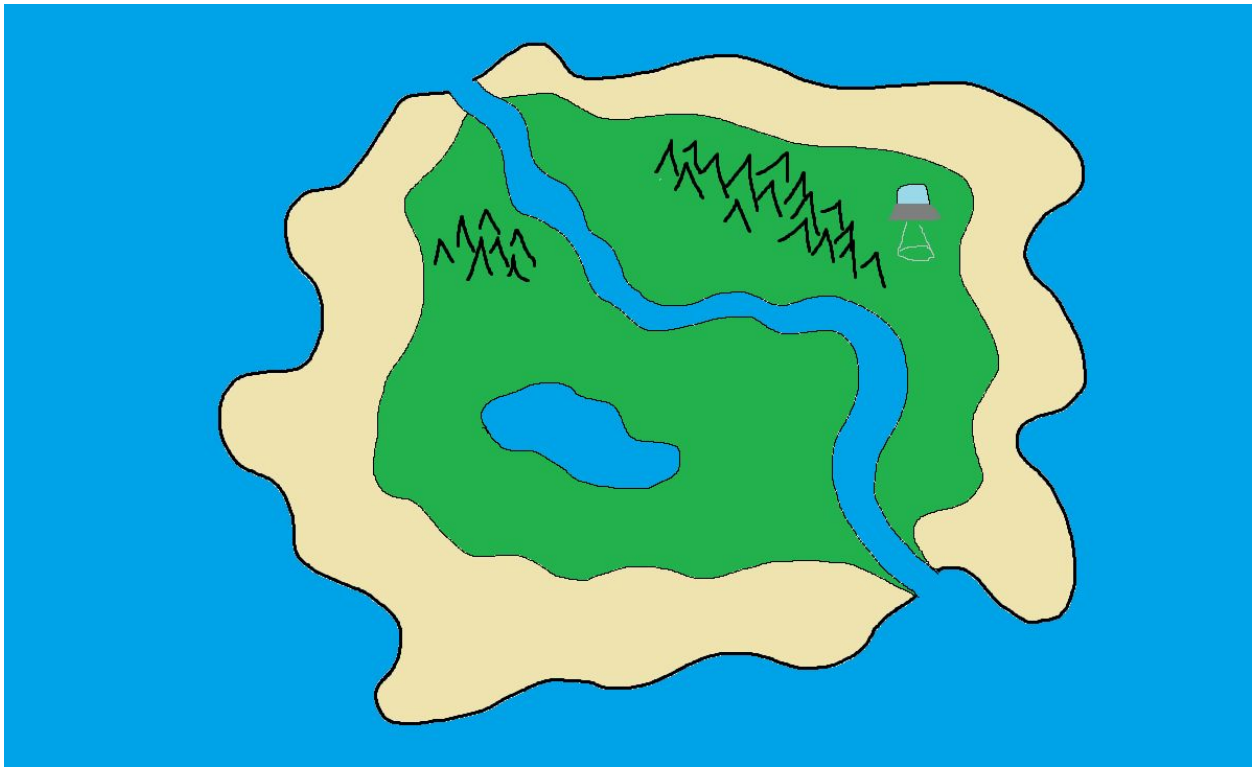

«interface»
Component
+ getName() const: string
+ getData() const: const ComponentData&
+ setData(const ComponentData&): void
Velocity
+ vel: Vector2i
Position
+ posi: Vector2i


InputManager
+ keys: unordered_map<Keyboard::Key, string>
+ state: unordered_map<string, bool>
+ associate(Keyboard::Key, string): void
+ activate(Keyboard::Key): type
+ deactivate(Keyboard::Key): type
+ isActive(string) const: bool
+ isActiveOnce(string) const: bool


<< Enumeration >>
Type
Texture
Font
SoundBuffer
ResourceManager
+ textures: unordered_map<string, Texture>
+ fonts: unordered_map<string, Font>
+ sounds: unordered_map<string, SooundBuffer>
+ currentManager: ResourceManager
+ load(Type, string, string): void
+ getTexture(string): Texture&
+ getFont(string): Font&
+ getSoundBuffer(string): SounBuffer&

# Level Maps

- Island outlines.
- Water as an obstacle.
- Mountains that are impassable.
- Some vegetation passable, other vegetation must be eaten.
- Dinosaur size a factor in determining passability.



# Mechanics Analysis

- Moving: player moves x number of spaces at a time when they choose to move. There is an acceleration component, with the dinosaur able to move more spaces each turn it is moving until it reaches maximum speed (determined by dinosaur type). Moving into food eats it, moving into enemies calculates combat, moving into powerups collects them. Some terrain may cost more energy to move through. Running into obstacles ends movement and forfeits remaining movement spaces for turn. Running into a powerup lets the player keep moving.
    - Scrolling:
        - Centered on player.
        - Have enough water around island that player can't see past edge of map.
- Combat: chance for defendant to flee and not receive damage (moves to next space in facing direction). Otherwise damage to each combatant will be calculated based on attack and defense. Special abilities and powerups that are toggled at the time combat begins will affect these attack/defense and thus the damage. If a combatant dies, it is removed and leaves behind a meat item.
    - Chance to flee: based off health remaining.
    - Player can flee by moving out of enemy space. Count as moving more spaces than usual to use more energy.
    - Dinosaur wait to chase player down after fleeing.
- Energy: all dinosaurs start off with same amount. Moving uses set energy amount, so faster dinosaurs use more energy by virtue of moving more. Special abilities will cost energy based on the type of ability. Can be refilled by eating. Some terrain may cost more energy to move through.
- Health: starting health based on dinosaur. Damage comes from combat. Healing through powerups. Slight healing through eating that can only reach 80% of maximum health. Slow regeneration rate that depends on difficulty and will be delayed after damage, and will only regenerate 5-10% above damage taken. If energy runs out, health will fall.
- Special abilities: attack, defense, and movement. Use up more energy than normal counterparts. Changes color of "halo" around dinosaur to show that it is active.
- Powerups: all powerups either positive or tradeoff. Combat (increase attack/defense/speed at cost of others), positive energy (boost), positive health (boost). Boost are immediate. Combat powerups last through next combat. Choose when to activate powerups. Limited inventory of powerups encourages player to think strategically. Max of two combat powerups activated at once also encourages strategic thinking.
- Spawning: certain amount of plants, herbivores, and carnivores generated at start. The escape pod will also be placed when the map is first populated at a location far from player. Randomize starting location. Plants regrow over time. Spawn a new dinosaur when one dies.

- Herbivore-herbivore does not lead to combat.
- Only one combat allowed at a time.
- AI dinosaurs only interact with player.
- End: death, finding escape pod, meteor.
- What counts as levels of success?
    - Death/Meteor: failure, no success.
    - Escape Pod: ranked based on turns taken, health, energy. Displays message on game over screen of how fit the aliens think you are to be saved. For carnivores, dinosaurs killed will be taken into account, while herbivores will gain favor for surviving being attacked.
- Meteor Timer: ticks down every turn. Turns left until meteor strike will be displayed in the upper left of the screen.
- Scoring: egg collection.
    - Eggs are placed like food and powerups.
    - Display on UI.
    - Determine whether aliens accept you.

# Scheduling

- Milestones:
    - *System stuff: 10/22 (Dan)*
    - *ResourceManager: 10/22 (Dan)*
    - *InputManager: 10/22 (Dan)*
    - *Map loading and Drawing: 10/22 (Ben) (Kevin: tile art)*
    - *UI Placeholder: 10/22 (Ben)*
    - *Entities, Components, and Systems: 10/22 (Cindy)*
        - *System base class and array in Logic.*
        - *Components: position and velocity.*
        - *Entity with component unordered_map. Also accessors for components.*
    - *Moving and Turn system: 10/22 (Kevin)*
    - Meteor countdown: 11/12 (Cindy)
    - ECS Updates: 11/12 (Dan)
    - Scrolling: 11/12 (Ben)
    - Dinosaur types and stats: 11/12 (Ben)
    - Placement (Enemies, Escape Pod): 11/12 (Cindy)
    - Energy and health systems: 11/12 (Kevin)
    - Food component and eating: 11/12 (Kevin)
    - Visual Component: 11/12 (Cindy)
    - Acceleration: 12/3
    - Terrain Movement Cost: 12/3
    - AI and combat: 12/3 (Everyone)
    - Enemy moving (Path Finding): 12/3
    - Powerup implementation: 12/3
    - Eggs: 12/3
    - Special abilities: 12/3
    - Sounds (including radar): 12/3
    - Menus: 12/3
    - Configurable Keys: 12/3
- Project Dates:
    - Design Presentation: October 3rd
    - Design Document + Report: October 10th
    - Intermediate Presentation & Report: October 22nd
    - Intermediate Presentation & Report: November 12th
    - Demo: December 3rd Final Presentations: December 5th
    - Final Report: December 6th

# Changelog

- Revision 1: initial version of all pages.
- Revision 2:
    - Add Entities, Components, Systems to schedule.
    - Move dinosaur types, energy + health systems, and meteor countdown to 11/12 milestone.
    - Removed EventManager from architecture and switch to direct logic-view communication.
    - Make map data one layer.
    - Added InputManager to schedule.
    - Clarified what running into obstacles does.
    - Decided on how collision will work.
    - Assigned first presentation tasks.
    - Added configurable keys to schedule.
    - Add tags for movement actions
    - Added diagrams for software architecture.
    - Added more details to related games.
    - Added Overview section.
    - Added diagrams for Software Architecture.
- Revision 3:
    - Define how scrolling will work mechanically.
    - Schedule scrolling.
    - Move path finding back to final presentation.
    - Add Thor as an external library.
    - Change spawn-death condition (spawn new when one dies, rather than occasionally spawning new ones based on a threshold).
    - Added EntityBuilder to architecture.
    - Defined specific systems in architecture.
    - Defined new update process in architecture.
    - Specified what visual effect special abilities have.
    - Added egg component.
    - Decided dinosaur components and file format.
    - Adjusted combat mechanics (changed flee decision, removed dodging).
    - Scheduled for second intermediate presentation.
    - Clarify dinosaur types as resources.
    - Add and update diagrams.
    - Move AI and combat back to final presentation.
    - Decide entity ranges.
    - Schedule visual component.