The Legend of Phil A. Buster

UW Tri-Campus Game Jam 2025

The Team:

James, Caeden, Dan, Wyatt, Prahas, Riley

Introduction

The Legend of Phil A. Buster is a roguelike, mad-lib-like, deck-building-like word game. You play as Phil Buster, a senator who must filibuster his way through each day, wasting time and delaying pesky bills and boring policymaking.

Similar games include *Oh…Sir! The Insult Simulator* for its mad-lib mechanics and *Balatro* for its deck building mechanics.

Gameplay Mechanics

Core gameplay mechanics include building two separate decks of cards containing a word: one deck of nouns and verbs, and another of adjectives and adverbs (used to modify nouns and verbs for bonus effects).

The player must filibuster their way through a series of increasingly challenging senate sessions. After each senate session, the player has an opportunity to spend money on building their decks at the shop.

The Senate Session

During a senate session, the player is dealt a random hand from their deck, and a random bill/policy is assigned for the day. The player must then fill in the empty fields in a given sentence using words from their hand, à la *Mad Libs* (i.e. noun fields must be filled in with a noun card, and verb fields with a verb card). Once a sentence has been completely filled, the player can then submit and the character Phil Buster will read the sentence. The amount of time wasted depends on the sentence length and possibly word syllables.

Each noun and verb card will be assigned a set dollar amount awarded upon playing, usually scaling with its rarity. A time wasted amount will also be assigned, usually corresponding with the number of syllables. Additionally, hidden scalers (on a range of -1.0 to 1.0) are assigned to a set of “bill concepts” (e.g. taxes, infrastructure, debt, social security), which is used to calculate the relevancy of that word to a session’s bill when played.

Optionally, adjectives and adverbs can be combined with noun and verb cards for additional effects, such as multipliers for money, time, and Bust Meter.

The player will then continue being dealt hands and random sentences until either the session time runs out (win state) or the player’s Bust Meter fills up (fail state).

The Deck

The player starts the game with a base deck of about 20-30 (exact details TBD) nouns and verbs, and a completely empty adjective and adverb deck. At the start of the Senate session, the player draws a number of nouns/verbs cards and adjectives/adverbs cards into their hand. Word cards that are played as part of a sentence, or cards that are discarded from the hand, are out of play until the next Senate session.

As the player purchases more cards from the Shop, they can remove and add cards to their deck by selecting them from a list of all nouns/verbs and adjectives/adverbs that have been unlocked. The player can have a maximum of 30 cards in the combined decks.

To determine: How to best draw nouns/verbs and adjectives/adverbs evenly? Do we just not bother and randomly draw from either deck? Or do we actually make some attempt at making sure the game pulls evenly from both decks, if possible?

The Bust Meter

The Bust Meter is a meter that gauges how “obvious” the player is filibustering. If the Bust Meter fills up, Phil Buster is busted for filibustering and the game immediately ends.

The Bust Meter fills up in multiple ways: the player plays an irrelevant word to the given bill at the time (e.g. playing the noun “visa” when the bill concerns social security); the player discards a selection of cards from their hand; the player submits a sentence with incomplete fields; the player runs out of time before submitting and is forced to submit an incomplete sentence.

The Bust Meter can be emptied in only one way: the player plays a relevant word to the given bill at the time (e.g. playing the verb “nuke” when the bill concerns foreign policy). This effect is usually small, but can be multiplied with adjective and adverb bonuses.

With each senate session, the Bust Meter decreases in size, decreasing the amount of leeway for mistakes.

The Shop

After each senate session, the player is able to purchase cards and build their deck for the following sessions. Using the money earned from playing certain cards during the senate session, the player can purchase from a randomly generated set of nouns and verbs, and another with adjectives and adverbs (smaller selection).

Optionally, the player can reroll the shop for a price. The player can also remove cards from their deck. The player can also decide not to purchase anything and save money for the next shop session.

The Shop can also be used to purchase temporary upgrades for the next Senate session. For example, Insider Knowledge will allow the player to know the topic of the next round. The player can also invest in their performance in the next round, earning more money if they reach a certain threshold.

Idea: player can buy upgrades like longer sentences, more nouns than verbs, maybe force a certain bill (high cost), force cards to be of a certain topic(?)

Post Game

The player will be able to view the transcripts of their senate sessions.

Gameplay Strategies

Player noun and verb cards will generally be categorized into three categories: high dollar value cards, high relevancy cards (usually generic words), and high time wasting cards (usually big words). During early senate sessions, the player may choose to prioritize high dollar value cards at the cost of larger bust penalties since the Bust Meter will be fairly lenient. Then later on as the Bust Meter shrinks with each senate session, the player may transition towards deck builds with either safer high relevancy cards, or high time wasting cards (perhaps depending on adjective and adverb cards purchased and their bonuses).

Monetization Strategy

All proceeds go to james.

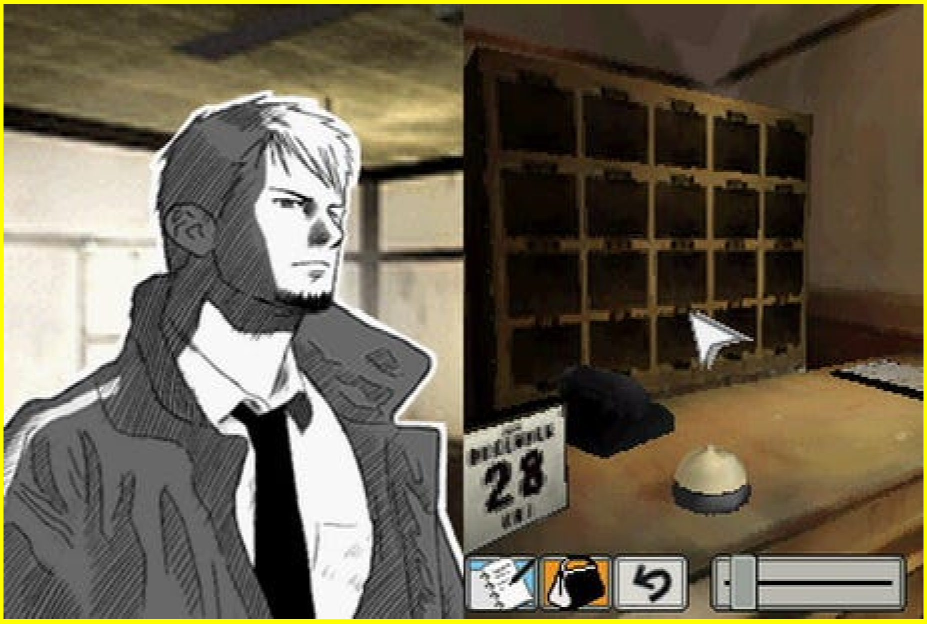
Artwork

IMPORTANT: viewport resolution is 1920x1080. Make sure your textures are to proper and intentional scale inside this viewport size.

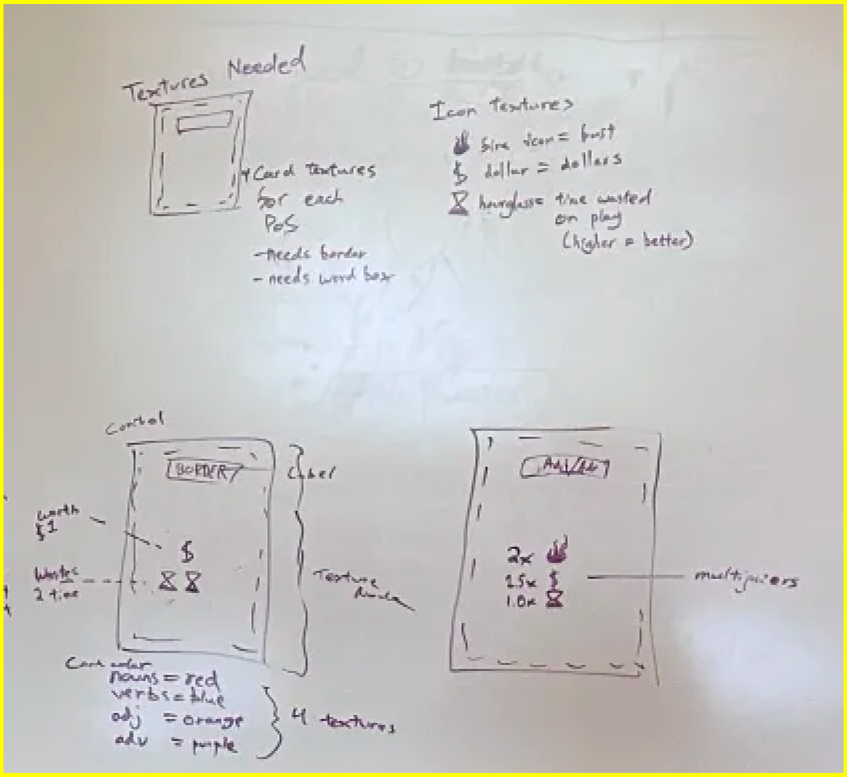
Background art style: Muted yellow-y colors

Characters: Black and white sketches a la Ace Attorney

TODO: Make a mood board here: just post images, color schemes, themes, etc. that can help out with design



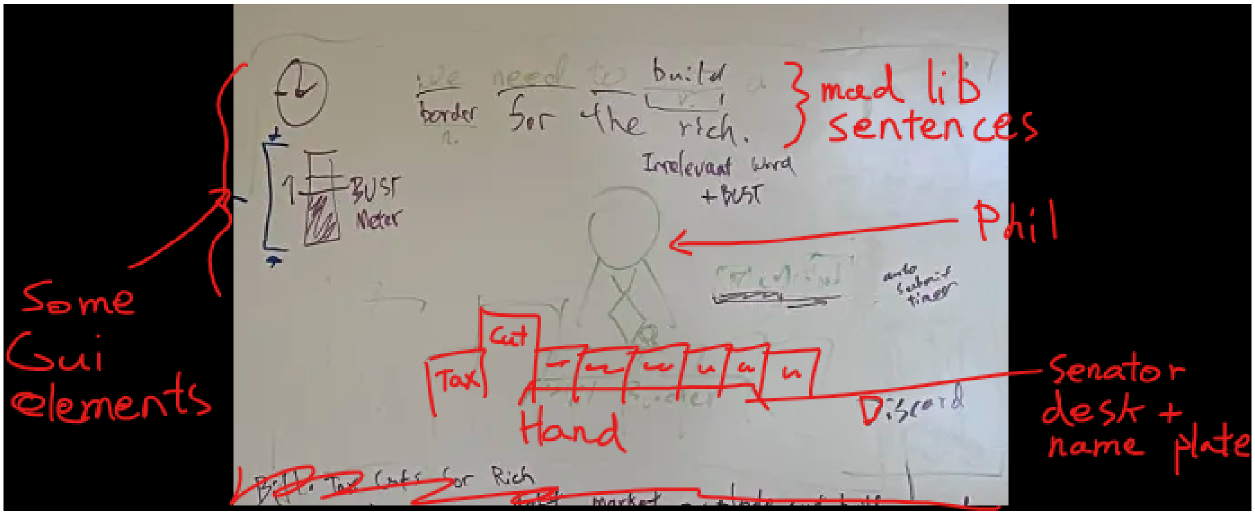
21 June 1956
Lucidity Times
The GREAT FOSSIL FETCH!
Name Your Prize



Cards will need four textures corresponding to the four parts of speech: noun, verb, adjective, and adverb. Noun cards should be red, verbs blue, adjectives orange (to correspond with red nouns), and adverbs purple (with blue verbs). Or pick a different color scheme, but should be with distinct warm colors and cold colors. The card border should also be unique for each card type.

Icons are needed for certain gameplay elements. Card price should be a dollar sign. Time should be an hour glass. Bust Meter icon needs some sort of icon, we’re not sure. Maybe a flame? Or a boot. Or a guillotine.

Relevancy will be a hidden mechanic for players to learn, so we won’t give it any UI presence.



GUI elements are needed such as a clock (representing session time) with separate, rotatable hand sprites.

A bust meter (base “container” with a tileable, square, and perhaps animated fill texture that is drawn on top, and a mask/overlay texture that masks the fill texture, allowing for cool shape or something).

Work Delegation

Prahas handles UI art once again. Unless dan/wyatt also want a crack at it.

Prahas or Wyatt or Dan or something (you guys can split the work somehow) do card art, icons, character sprites

Writing can be Dan or everyone

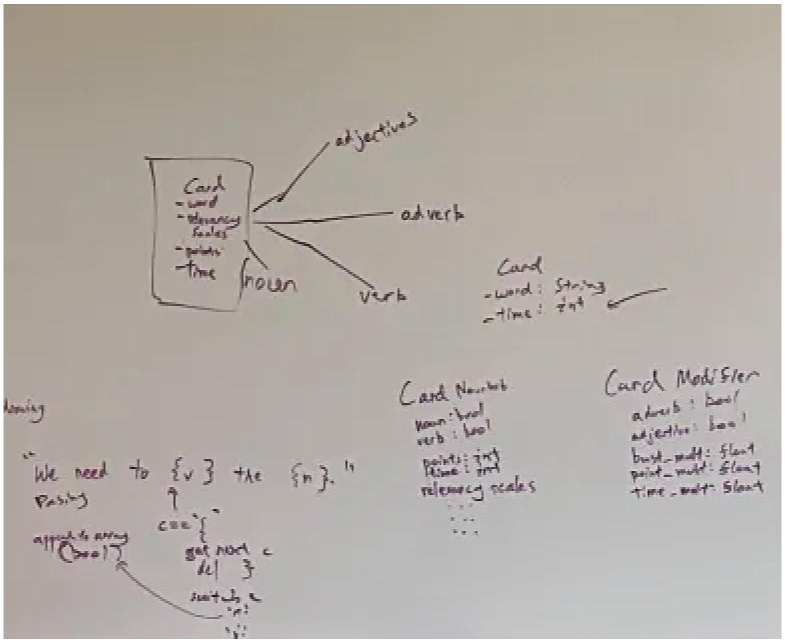
Don’t worry about making every word/sentence combination grammatically correct - silly incorrect sentences might as well be part of the fun.

Word design and balancing (relevancies, values, and time) should be done later.

James and Caeden with coding. By tomorrow, basic gameplay loop will be done.

Code Design Notes

James: I’ve already got plans for the sentence prompt (container with added control elements for each word/field, allowing for dynamic fitting and wrapping), card dragging (basically copying code from scrabble game), shop (from learn to swing)



Base class Card extends Node2D

- word: String
- rarity: int ??? could allow for some special particle effects

CardNoun extends Card

- dollars: int
- time: int
- relevancy scales (multiple): float (with export_range(-1.0, 1.0, 0.01, 0.01))
- e.g. relevancy_immigration, relevancy_tax

CardVerb extends Card

- points: int
- time: int
- relevancy scales (multiple): float (with export_range(-1.0, 1.0, 0.01, 0.01))
- e.g. relevancy_immigration, relevancy_tax

CardAdjective extends Card

- point_multiplier: float
- time_multiplier: float
- point_multiplier: float

CardAdverb extends Card

+ point_multiplier: float
+ time_multiplier: float
+ point_multiplier: float

Ok i had a thought: rather than deal with words that can be either noun or verb (in which case, what card texture to use? Some mix of color? Now thats confusing. Also may complicate code a bit idk), we just force the player to do with it lol. “Drive” can technically be noun or verb, but we decide which one for that card 🙂

Caeden: i have ideas for the card / sentence data structures and might start work on that (we briefly discussed generating sentence resources from a text file)

Can yo udesign a system for reading/storing sentences? I think we talked about using a spreadsheet, but i think a simple text file will do (or multiple text files)

Also should each topic have its own set of sentence structures? Maybe there’s also a generic set of sentence structures that can be used for all topics.

I think a text file for each topic will do (I will also consider adding a generic topics list that gets added into every topic list)

Make a class (extends Node? Or Resource? Not sure, doesn’t really matter) that will read all topic text files to arrays on initialization _init(). Make a folder for storing the text files, have a generic.txt for the generic list, and then all the other text files be named to their topic. The class can then read all text files in the folder and make a Dictionary[topic: String, sentences: PackedStringArray]

then the main game loop can ask for a topic‘s random sentence string via a function like get_topic_sentence(topic: String) -> String

We may also want support for a filter for longer sentences. On file read, binary insertion based on word count for each sentence?

James: [DONE] Proper sentence processing will be done by the sentence prompt node, which will instantiate Label controls with the sentence words, and replacing {n} and {v} with the noun and verb fields (custom Control script). Each word needs to be a separate control, since the sentence prompt will be inserting controls in a wrapping horizontal flow container.

TODO:

Note: must use mouse_entered/mouse_exited signals (get mouse positions dont check for other controls overlapping)

Drag system (cards are the only draggable objects). Draggable reads input + mouse hover -> emits signal on drag start/stop -> card_instance relays via signal to session.gd. session.gd needs to “turn on” relevant drag_drop areas (relevant as in a noun card dragging does not highlight verb field). Maybe a set_highlight_noun() or set_highlight_verb() on sentence_container.gd.

Or card_instance has a “field” form

Field hover should display a card preview (hence field should handle instance reparenting)

For dragging cards out of a field: field also has drag_grab.gd, on start -> signal to sentence_container.gd

Audio/Music

A card “dealing” sound effect when added to the hand, like being flipped/flicked

A discard sound effect, like throwing a crumbled paper into a small plastic trash bin maybe?

Two different sounds for picking up a card, and placing a card down

Shop sounds: purchasing an item, like a cash register or coins dropping.

Invalid sound effect (e.g. when the player tries to put a noun card into a verb slot) that

A repeatable man voice sound clip (like undertale/omori) as text is read when submitting a sentence. engine can also pitch shift for variation. Go wild here i dont have anything specific in mind. Maybe wait for Dan to sketch the character

Some bust meter sound effect when it fills up. This is the fail state. I have no clue what it should sound like, maybe an alarm or bell, or could be a short tune

UI button clicks (we’ve yet to work on the UI so not sure yet)