Testing skeleton

Version: 1.1

Godot Unit Testing (GUT) framework for game unit testing:

```
func test_create_player():
    # Ensure a player object is created without any issues.

func test_gameboard_generation():
    # Ensure a game board object is created without any issues.

func test_select_cards():
    # Ensure a player can select a card from the game board.
    # Players can select 2 cards per turn.

func test_card_pair_match():
    # Ensure the game properly reports a match if 2 card selections
    # contain the same fruit or vegetable.

func test_game_results():
    # Compare card pair totals for each player (multiplayer) and report
    # the highest count as the winner or tie if there is an equal count.

func test_get_card_description():
    # If the player has a card pair, let them view the description of
    # the fruit or vegetable.
```

Selenium WebDriver for web interface unit testing (using Python 3):

Prerequisites: python3, pip package manager, selenium module, chromedriver, and import statement below

```
from selenium import webdriver


def create_browsing_session()
      """ ensure main page is accessible """
      browser = webdriver.Chrome()
      browser.get(link_to_website)


def open_game_page()
      """ ensure game page is accessible """
      browser = webdriver.Chrome()
      browser.get(link_to_website)
      browser.find_element_by_id(<game page button>).click()
      browser.implicitly_wait(number_of_seconds)


def open_payment_page()
      """ ensure payment page is accessible """
      browser = webdriver.Chrome()
      browser.get(link_to_website)
      browser.find_element_by_id(<payment page button>).click()
      browser.implicitly_wait(number_of_seconds)
```

Game integration tests:

```
extends "res://addons/gut/test.gd"
class TestPlayerClass:
	extends "res://addons/gut/test.gd"
	func before_all():
		gut.p("----Player Integration Test----")
	.
	.
	.

	func after_all():
		gut.p("-------------------------------")

extends "res://addons/gut/test.gd"
class TestCardClass:
	extends "res://addons/gut/test.gd"
	func before_all():
		gut.p("----Card Integration Test----")
	.
	.
	.

	func after_all():
		gut.p("-----------------------------")

extends "res://addons/gut/test.gd"
class TestGameBoardClass:
	extends "res://addons/gut/test.gd"
	func before_all():
		gut.p("----Game Board Test----")
	.
	.
	.

	func after_all():
		gut.p("-----------------------------")
```

```
# Big bang approach: load multiple files and make sure they all pass
extends SceneTree
func _init():
    load('res://scripts/unittest.gd').run([
        'res://scripts/player_unit_tests.gd',
        'res://scripts/card_unit_tests.gd',
        'res://scripts/game_unit_tests.gd'
    ])
    quit()
```

Payment system integration test:

```
class TestPaymentSystem(unittest.TestCase):

      """Run the series of unit tests with big bang approach."""

      def test_open_payment_page(self):

            """IMPLEMENT LATER"""

      def test_select_charity(self):

            """IMPLEMENT LATER"""

      def test_make_payment(self):

            """IMPLEMENT LATER"""

      ...
```

Web interface integration test:

```
class TestWebInterface(unittest.TestCase):

      """Run the series of unit tests with big bang approach."""

      def test_open_home_page(self):

            """IMPLEMENT LATER"""

      def test_open_game_page(self):

            """IMPLEMENT LATER"""

      def test_open_payment_page(self):

            """IMPLEMENT LATER"""

      ...
```

Regression test procedure:
Highest priority at the top, lowest priorities at the bottom.