

01 Game Design

Contents

[Overview 3](#_Toc506741905)

[1.1. Core Concepts 4](#_Toc506741906)

[Game Development Basics 4](#_Toc506741907)

[Game Design 4](#_Toc506741908)

[Player Motivations 4](#_Toc506741909)

[Documentation 5](#_Toc506741910)

[1.2. Common Mistakes 5](#_Toc506741911)

[Knowing When to Abandon Features and Mechanics 5](#_Toc506741912)

[Punishment versus Negative Reinforcement 5](#_Toc506741913)

[Organic Documentation 5](#_Toc506741914)

[1.3. Supplementary Resources 5](#_Toc506741915)

[Exercises 6](#_Toc506741916)

[Exercise 1A: Discussing Game Design 6](#_Toc506741917)

[Multiple-Choice Quiz 7](#_Toc506741918)

# Overview

**PURPOSE**

The purpose of this tutorial is to outline the basics of game design.

**SCOPE**

This section will provide an introductory overview of the following:

* Game design and development

**PREREQUISITES**

There are no prerequisites for this section.

## 1.1. Core Concepts

### Game Development Basics

The game development life cycle involves many stages that proceed in a **continuous cycle over time**. The design phase is the first phase, but over time **mechanics are revisited again and again** with fresh design changes until they perform in a way that works toward the overall design.

|  |  |
| --- | --- |
| C:\Users\Melanie Nikdel\AppData\Local\Microsoft\Windows\INetCache\Content.Word\tips-png-4.png | Sometimes these design changes can be substantial. For instance, *Diablo* by Blizzard Entertainment was originally designed as a turn-based game; however, after the concept was developed, tested, and evaluated, the game was switched over to a real-time project. |

### Game Design

Anyone can describe a game, but describing a game in a **game design context** is important for analyzing and critically comparing projects and mechanics to identify which elements can be further developed or expanded upon for game variation. A game of tag can be extended by introducing a ball as an actor and changing the tag mechanic to involve tagging with the ball for a substantially different game.

|  |  |
| --- | --- |
| C:\Users\Melanie Nikdel\AppData\Local\Microsoft\Windows\INetCache\Content.Word\tips-png-4.png | The Bird of Prey demo is used to help provide an understanding of the different elements involved in game design, but the logic can be applied to any game. Smaller games are better for critically analyzing since they involve fewer elements. Consider mobile games like *Fruit Ninja*, *Infinity Blade*, or *Bejeweled.* |

### Player Motivations

* Punishment is the most effective way to ensure that players cease to perform an action; however, once the punishment is removed, players are quick to adapt their behaviors. Games like *FarmVille* and *Tamagotchi* use punishment by destroying the players’ farm or killing their pet when they stop playing.
* Positive reinforcement at **random intervals** produces the longest delay in players’ altering their behavior. For instance, if a player is randomly awarded a power-up when they defeat an enemy, you could stop giving them power-ups and they’d continue their behavior for a longer period of time before they acknowledged that the rewards were no longer being awarded.

|  |  |
| --- | --- |
| C:\Users\Melanie Nikdel\AppData\Local\Microsoft\Windows\INetCache\Content.Word\tips-png-4.png | If you’d like to delve further into player rewards, you might like to consult B. F. Skinner’s research on operant conditioning. The Skinner box provides a reasonably clean example of the effects of positive/negative reinforcement and punishment with results that can be extrapolated to gameplay. |

### Documentation

* Different documentation serves very different purposes. It’s wise to make sure that only outward-facing documents, like pitch documents, are sent to potential publishers and not inward-facing documents unless explicitly requested.
* Documents need to be organic and evolve with the project. It’s common for large concepts to not make it into the final release due to development constraints or them simply not working.
* A feature matrix is a large document, but it can help uncover design flaws in a project well before implementation.

## 1.2. First Projects for Students

### Guidelines

As most student teams lack experience in game development or using Unreal some general guidelines for a first student project are:

* Limit the scope of the project.
* Keep the project simple and achievable.
* Make sure that there are references for any mechanics to ensure the project is achievable. Making something unique and quirky is great, but it’s certainly not essential for a first project.
* Try and get a diverse mix of team members to ensure that different roles are catered for. Often friends share similar interests and teams end up being exclusively made up of programmers or exclusively artists which is detrimental to a final student deliverable.

### First Project Ideas

#### Platformer

Developing a simple platformer game in the style of Crash Bandicoot or Mario is a simple way of developing straight forward mechanics and leveraging Blueprint for simple gameplay additions such as jump pads or elevators.

Consider referencing from the Unreal Launcher:

* Platformer Template
* Platformer Game Sample
* Content Examples (for contraptions)

#### Puzzle Game

A simple board game style project often doesn’t require much Blueprint or heavy asset development and can be easily cleaned up and refined by a small team.

Consider referencing from the Unreal Launcher:

* Match 3 Sample
* Radial Impact (for 2D effects)

#### Adventure Puzzle Game

Developing an experience focused on puzzles within a 3D environment allows for the project to be modular and have a more definable scope.

Consider referencing from the Unreal Launcher:

* First Person Template
* Blueprints Example
* Content Examples

#### Third Person Action Game

Beat ‘em up’s can be developed with limited scope can be great for focusing on pawns and animation systems for a first project. When approaching a project like this be aware of limiting the number of enemies & special moves as each one can slowly complicate gameplay and increase the art budget.

Consider referencing from the Unreal Launcher:

* Third Person Template
* Couch Knights (for combos and animations)
* Particle Effects Demo

#### Strategy Game

A top down strategy project in which units can be selected and moved to perform actions in the style of XCOM or Command and Conquer is a good way to learn how different types of controllers and pawns work together through the game framework.

Consider referencing from the Unreal Launcher:

* Strategy Game Example
* Turn Based Strategy Example
* Shooter Game (for effects and Pawn animations)

## 1.3. Common Mistakes

### Knowing When to Abandon Features and Mechanics

Sometimes when developing a project, a feature or mechanic goes through several iterations and still isn’t compatible with the overall game design. At some point, you need to ask whether the feature has the ability to still add to your project or if it should be left in the dust. Features may also be cut due to development-time issues. If something is going to take six months to develop, it’s potentially taking six months of development time away from other equally or more important features.

### Punishment versus Negative Reinforcement

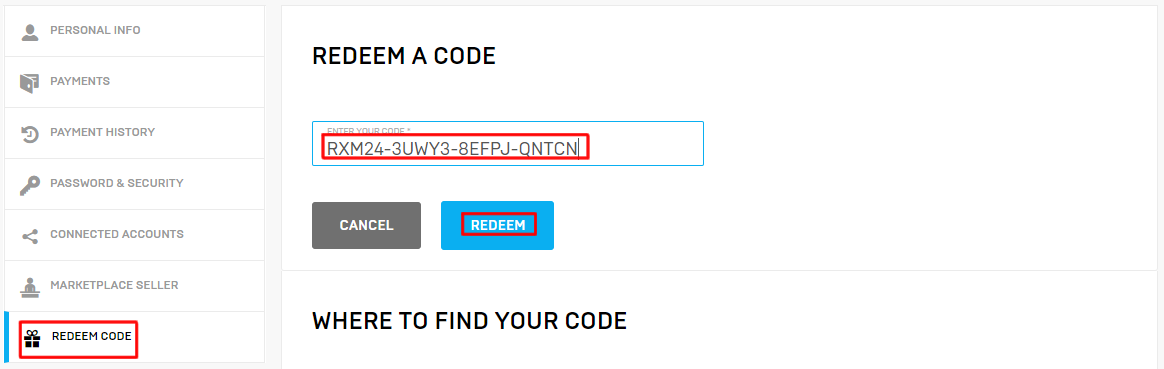
Punishment is used to decrease a behavior through the use of a negative stimulus, whereas negative reinforcement is used to increase a behavior. If housemates are constantly complaining that you must do the dishes and you do them to take away the nagging, the complaining would be an act of negative reinforcement.

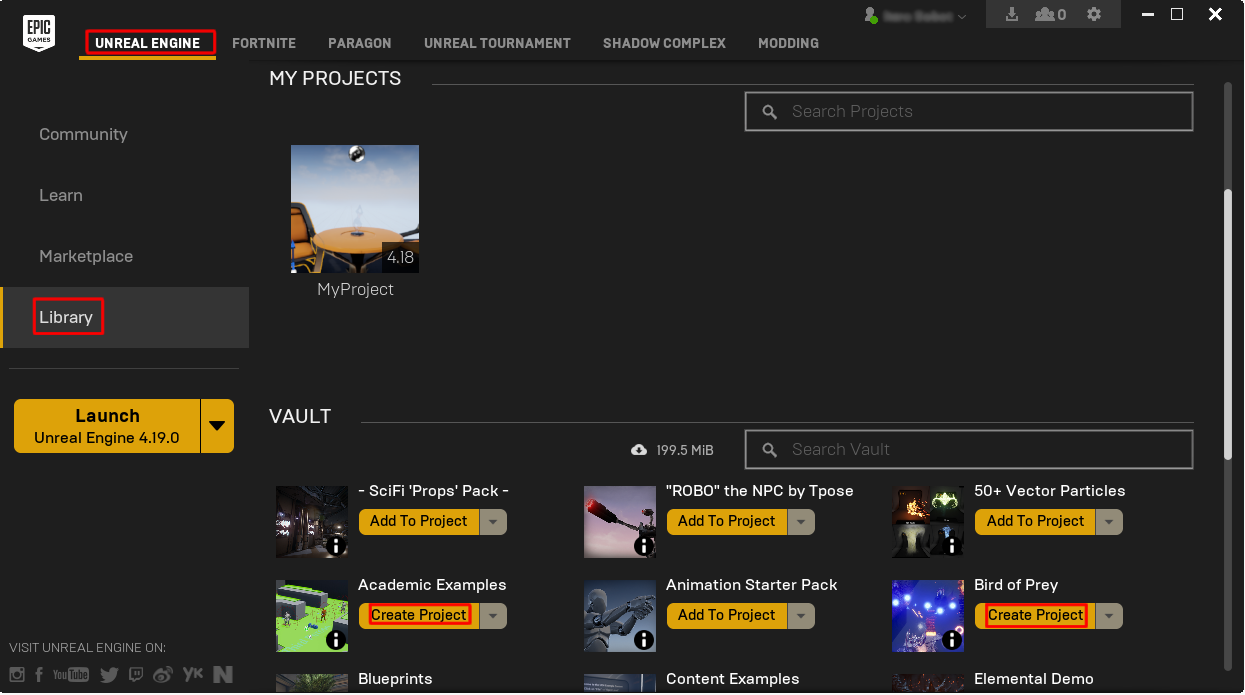
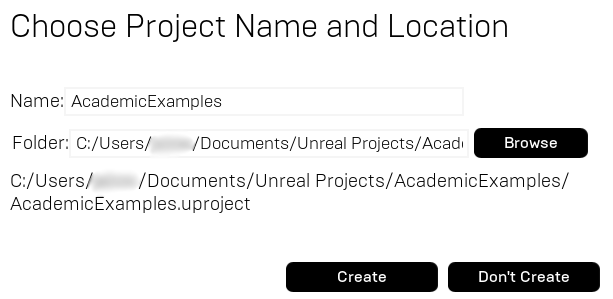
### Organic Documentation

Documentation being organic doesn’t mean a team should arbitrarily ignore the design document. It just means that when things need to change, the document should be updated to reflect the changes in order to ensure there’s clarity in development of the project.

## 1.3. Supplementary Resources

You will need these files to complete the Exercises on this page. These files are hosted on the Unreal Engine Marketplace and can be accessed from your Launcher by following the directions below.

1. On the [Code Redemption tab](https://www.unrealengine.com/dashboard/code-redemption" \t "_blank) (<https://www.unrealengine.com/dashboard/code-redemption>) of your Epic personal dashboard, enter the code RXM24-3UWY3-8EFPJ-QNTCN, then click REDEEM and CONFIRM  


2. Open the Epic Games Launcher (you can download the [Launcher](https://www.unrealengine.com/download" \t "_blank) [here](https://www.unrealengine.com/download)if you need it), navigate to the Vault section of the Unreal Engine Library tab, and click Create Project on Academic Examples or Bird of Prey  
  
  
3. Choose a project name and location and click Create  


And that's it! If you have any further questions about how to download these products from the Epic Games Launcher, please contact [marketplace-support@unrealengine.com](mailto:marketplace-support@unrealengine.com" \t "_blank).

# Exercises

## Exercise 1A: Discussing Game Design

**Deliverables:** In-class observation

Since this class will be the first to cover game development, it is recommended that you initially foster class discussion that focuses on the following topics and questions. This exercise is based on game development concepts, and since it’s likely that any class will include students with variable levels of understanding of these concepts, a group discussion before a quiz is employed.

**Directions and Questions for Class Discussion**

1. Discuss projects with the class in a **game design context**. If you’re concerned about the difficulty, then consider using smaller, more contained examples like *Angry Birds,* *Bejeweled*,or *Tetris*. Identify the actors, mechanics, goals, space, and rules for each project.
2. What small changes could be made to the concepts discussed? Which category, or categories, would these changes fall under?
3. What is the difference between punishment, positive reinforcement, and negative reinforcement?
   1. Name some games that use positive reinforcement and to what effect.
   2. Name some games that use negative reinforcement and to what effect.
   3. Name some games that use punishment and to what effect.
4. What are the differences between a design document and a pitch document? When hiring new staff, which document would you potentially expose them to in the early stages of a project?
5. What elements would be involved in a feature matrix for a game of hide-and-seek?
6. Why would an art bible be necessary?
7. Consider a budget for a small indie team. What would it look like? What unexpected costs could arise?

# Multiple-Choice Quiz

|  |  |  |  |
| --- | --- | --- | --- |
| Name: |  | Date: |  |

1. In a game design context, players are
   1. Mechanics
   2. Goals
   3. Space
   4. Rules
   5. Actors
2. The act of tagging players in a game of tag is a mechanic.
   1. True
   2. False
3. A player loses their health every time they hurt a friendly player in a game. In terms of operant conditioning, this is known as
   1. Positive reinforcement
   2. Justice
   3. Negative reinforcement
   4. Punishment
4. A design document should be given to prospective hires.
   1. True
   2. False
5. Indie teams don’t need to do production budgets because their costs are so low anyway.
   1. True
   2. False
6. Design documents should be organic and change over time.
   1. True
   2. False
7. What you originally design is always what you ultimately release.
   1. True
   2. False
8. A document in which you specify the poly counts, number of LODs for specific assets, and instruction count limits is known as a(n)
   1. Pitch document
   2. Production budget
   3. Design document
   4. Art bible
   5. Art budget