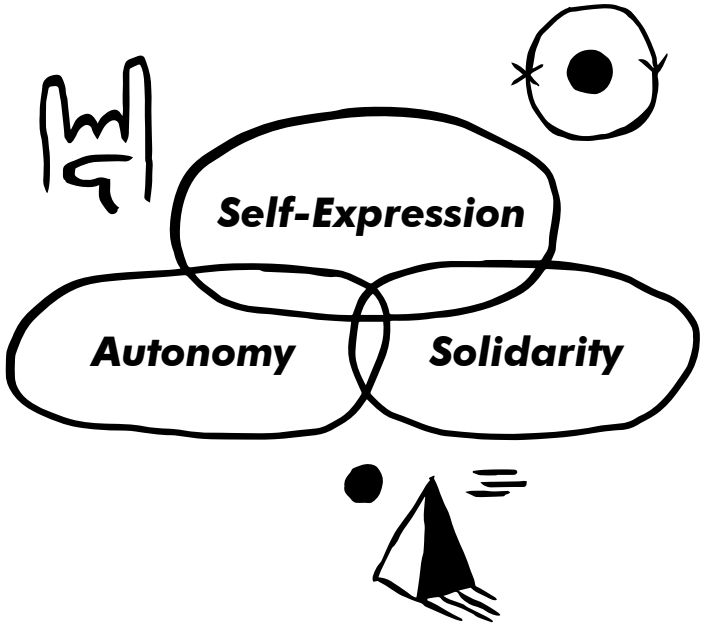
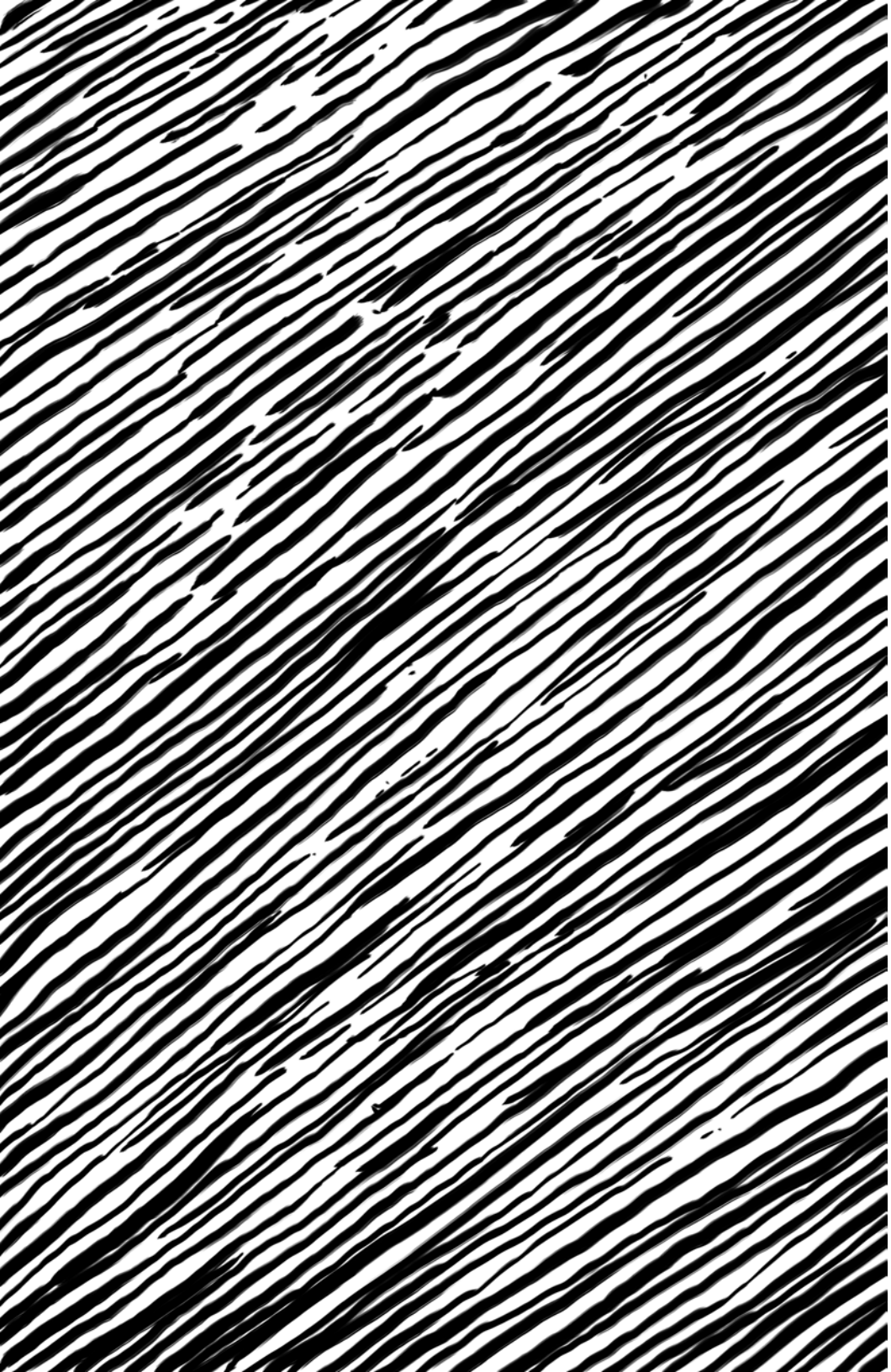
Radical Play

Sample Curriculum





Overview

This is an example curriculum for an introductory course on video game design and development for youth ages 12 and up, with little-to-no prior game making experience.

Its explicit objective is to expose students to a variety of design methodologies, diverse array of game development software, and novel game play experiences to encourage student *self-expression* through video game design.

This course will also aim to develop students sense of *autonomy*, by using a variety of classroom management techniques, and allowing students to choose their game-making tools and creative path.

The schedule is structured to allow for a period of mutual acquaintance between students and instructors; this curriculum must be adapted on a case-by-case basis taking into consideration the instructors' and students' skill set and background, as well as the class' shared values.

The instructor's main task is to create a context in which students will motivate themselves and pursue their interests with the help of peers and a caring adult.

Course Goals

- Expose students to a wide range of game-creating tools and design processes, including analog games and prototyping.
- Expose students to a diverse range of play experiences.
- Empower students who are traditionally discouraged from participating the field (non-male, non-binary, students of color, 1st generation) to make games.
- Each student makes at least one playable video game of their own by the end of the course.
- Expose non-programmers to some programming, non-artists to some art-making, non-writers to some writing, etc.
- Prepare students for self-guided learning outside of the classroom.
- Contribute to students sense of autonomy, enable self-expression through a making practice, in an environment that fosters solidarity.

What are *not* goals of this course

- Teach programming.
- Teach students to become experts in only *one* specific engine or software.
- Prepare students for a career in game development.
- Guide students in the making of a complex game with intricate graphics, sound, and animation.

Daily Class Structure

- 1 hour of show & tell, tutorials, lectures, etc.
- 2 hours of uninterrupted work time, minimum.
- Work time increases as class moves forward, as students get comfortable with skills and software, and need time to work on their games.

Class Organization and Composition

- One or more main instructors that act as guides for the learning process. They have experience on the field and share their subjective accumulated library of games and game-making skills, helping expose students to new experiences. This needs to happen by setting the context for learning, not by authority.
- One or more assistants. Assistants work together with instructors in guiding and sharing, but have less responsibility in structuring curriculum and activities.
- Student body must be composed by *at least* 1/2 of under-represented demographics (i.e. women, people of color, queer students, ESL students, or a combination). If the class is composed almost entirely of PoC students (for example), then at least 1/2 must be non-male.
- Instructors, assistants, and students work together on games and class activities. Assessment happens regularly to determine if students are having their needs met.
- Syllabus is constantly altered during the course based on input from students or a perceived stronger interest on a particular subject or tool.

- Students who are interested in learning a completely new skill that facilitates game making (e.g. programming, 3D, writing) will work closely with instructors and assistants in specialized workshops.
- There is *no hierarchy of skills* in this class, e.g. programmers are not better game designers than writers.

Tools and Guidelines

- Don't use competition or rely on grade assessment.
- Introductions: Have students pair up, introduce themselves to each other, and then introduce their partners to the rest of the class.
- Rotate groups often. A random group generator is helpful.
- Practice *Pair Programming*: Students pair up and take turns driving (telling the partner what to type) or navigating (writing the code). Switch after a few lines.
- Use the *Ask 3 Rule*: Ask for the help of 3 people (the internet counts as 1) before relying on the instructor.
- Encourage *Step up/Step back*: Those who speak a lot step back, those who are quiet can step up.
- Limit lecture time, increase "play" time with tools.
- Review values list with students every 1–2 weeks.

Schedule

- Week 1: Introductions and sharing
- Week 2: Intro to tools and design frameworks
- Week 3: Assets (graphics, sound, etc.)
- Week 4: Game proposals, workshops
- Week 5: Development time
- Week 6: Polish, play testing, presentation skills

Course Outline

1. Class Values

- Compile list of values to be used as guidelines for syllabus changes and classroom conduct. Use a process of consensus and keep the list to a manageable number.

2. Game Design Basics

- Class discussion: what is a game?
- Brief history of games
- What can you say with a game?
- Playing as a game designer
- Game design methodologies
- Modifying (or "modding") existing games
- Writing game rules
- Paper prototyping and play testing

3. Video games today

- Survey of the state of the medium: from short authorial games, to mobile games, to AAA blockbusters
- Meta-game: modding, Let's Play, eSports, speed-running
- Issues: identity, politics, education
- Research: artificial intelligence, procedural content generation, games and education
- Game jams
- Publishing

4. Game development skills

Workshops and tutorials organized in smaller groups, depending on student need, skill levels, and interest.

- Intro to programming
- Intro to 2D and 3D art making and art asset resources
- Sound editing and resources
- Basics of animation
- Writing for games
- Collaboration and skill-sharing

5. Game development software

- Alongside each software, students will learn how to look for resources for self-guided learning outside of the classroom.
- Engines: Twine, Unity, Other engines (GameSalad, GameMaker, Javascript/HTML5 engines, Unreal)
- 2D art software: Photoshop/GIMP, Illustrator/Inkscape
- 3D art software: Maya/Blender, ZBrush/Sculptris
- Audio editing software: Audacity

6. Making the first game

Students will develop a game idea, individually or in groups, and will pursue it to completion in their preferred game engine with guidance from instructors.

Topics covered:

- Idea development
- Proper scoping
- Paper prototyping
- Prototype in selected engine
- Asset creation
- Play testing
- Troubleshooting
- Publishing

Example Final Assignment: *Game as Gift*

Design and develop a video game as a gift for a person you care about. Think of the game as if it was a letter or birthday card; you will be designing it specifically for that person. What do you want them to think of or feel when they play it? How is it relevant and personal to your relationship with them?