12 THINGS EVERY VIDEO GAME HAS

CoderDojo

Produced by the CoderDojo Foundation // @CoderDojo

1 Winning

Most games have a **goal**: some way of winning and, usually, seeing a **"You Win!"** message on the screen. Without a **goal**, a **player** doesn't know when they're done and they never have the fun of knowing they've beaten the game.

For example, in platform games (like **Mario** or **Rayman**) your **goal** is to reach the end of the stage. In other games your **goal** might be to collect the most points before time runs out, or before you lose all your health.

There are a few games with no clear built-in **goals**, like some versions of **Minecraft**. They instead offer a world where the **player** can explore and decide their own goals. These games are harder to build and harder to make fun for the **player**.

Start planning your game!

Have you decided on a **game** you want to make? Then take a piece of paper (or a file on your computer) and write down the answers to these questions:

- Does your game have a **"You Win!"** screen?
- How does a **player** reach that **"You Win!"** screen?
- Is the **goal** of the game easy for the **player** to learn and understand?
- Does the **player** know what they need to do to reach the **goal**?

CoderDojo

Produced by the CoderDojo Foundation // @CoderDojo

2 Losing

Most games also have a **"You Lose"** screen, when the **player** loses all their health or fails to complete the **goal**. It might also be a **"Game Over"** or **"Time's up"** screen depending on how you've decided your game works. Without a way of losing there is no risk for the **player**, no uncertainty in completing the **goal**.

Decide what **resources**, like health, magic power, tools or fuel, that the **player** can use to complete the **goal**. A very easy way to add a losing condition is by adding a **time limit**.

Other **"You Lose"** screens could be events: When the **player's character** falls down a pit, or when a customer in the **player's** restaurant loses her patience while waiting for her dish.

In most games, if you start the game and do nothing, you eventually lose. This pushes the **player** forward and encourages them to do something, adding excitement.

Also, you should decide how long a game will be. If a **stage** or **level** takes two hours to complete, it might be too long for a simple platform game. Be careful though: The **player** might not feel they've succeeded in anything if it only takes a couple of minutes to complete.

Update your plan!

Get your plan out again and note down the answers to these questions:

- What can cause the **player** to lose the game?
- What can the **player** do to avoid losing?
- What is the ideal game length, if the **player** plays well?

12 THINGS EVERY VIDEO GAME HAS

CoderDojo

Produced by the CoderDojo Foundation // @CoderDojo

1 Objects and variables

Every game has pieces that combine with other pieces to create an experience. For example, a racing game might need cars, roads and scenery around the roads. It might even need a street, street lights, a sky, the sun or the moon and stars.

You need to decide on **objects** your player **can see** in the game world. You also need to decide everything your player **cannot see**. For example the car might have a current speed displayed on the game screen, or it might be hidden from the player. The game might have time running out, or an inventory where the player has or doesn't have specific items. The player might have a certain amount of health. Once you have defined everything your game needs, you will need to bring it into your game. **Objects** will be shapes or images that might have animation, unseen information will be **variables** that your game saves and makes calculations with.

Update your plan

Get out your game plan again and update it with these questions and answers:

- What **objects** exist in the world I am building?
- What unseen values (**variables**) exist in the world I am building?

CoderDojo

Produced by the CoderDojo Foundation // @CoderDojo

2 Obstacles

You have defined your game's **goal** and what **objects** are in the game. It's time to use the rest of the game's components to create **obstacles** that make it harder to reach the **goal**. Without **obstacles**, the player can simply get straight to the **goal** with one button.

Obstacles can be moving, like enemies, or unmoving, like boxes to jump over. They can have a specific pattern of movement, for example up and down, or might be affected by the player, for example always moving towards the **character**.

Each **obstacle** should ideally be introduced in the game separately, so the player can learn how to face one thing at a time. Make early parts of the game easier by having fewer **obstacles** or having to face them less often. Later parts of the game can have more obstacles or combinations of different obstacles.

These obstacles are going to require skill from the players. They will have to figure out what they can do to get past them and what makes them **lose** the game. Then they will have to practice how to beat those **obstacles**, and through practice they'll get better at facing them.

Update your plan

Get out your game plan again and update it with these questions and answers:

- What is going to stop the player from **winning**?
- How each of those **obstacles** behave in the game?
- What can the player do to overcome the **obstacle**?

12 THINGS EVERY VIDEO GAME HAS

CoderDojo

Produced by the CoderDojo Foundation // @CoderDojo

1 Player Controls

Every game has **controls** because games are interactive. Without controls you might be building a movie or an image.

You need to decide which components the player controls in the game and which they don't. For example the player may control a car, but not every car on the street.

Once you decide the actions the player can take, you need to decide the input that is linked to each action. You can use different hardware to input **controls**, for example a keyboard, a mouse, a gamepad, a smartphone's motion detectors, even a camera.

The keyboard arrows could move your **character** around, the tilting right of the phone could steer your car to the right. You will need to define what each button or other **control** will do, if anything. For example, a platform game might have the character jump with **Space**, move left with **Left Arrow** and move right with **Right Arrow**.

Update your plan

Get out your game plan again and update it with these questions and answers:

- What **objects** and **variables** does the player control?
- What can they do with them?
- Which **control** does each action?

CoderDojo

Produced by the CoderDojo Foundation // @CoderDojo

2 Interactions

A game has to have **interactions** between the components the player controls and the ones they do not control. Without **interactions**, the player is a ghost walking through walls and cannot change anything in order to achieve the game's **goal**.

Decide what happens when two different **objects** meet. Maybe the **character** can push the brown box back but the black box is too heavy to move. Maybe an **object** changes, for example a balloon meeting a needle changes to a breaking balloon with a breaking sound then disappears. Maybe an enemy touching your character makes you lose health (a **variable**) and become invulnerable for two seconds.

There might be **interactions** between two **objects** that the player does not control. For example an enemy might be unable to jump or might fall down a pit if player lures him there. Try to make your **interactions** as natural as possible. When a needle meets a balloon, no one expects that the needle will explode.

Update your plan

Get out your game plan again and update it with the following:

- A list of all combinations of two **objects** that can happen your game and how they will **interact**
- Each **interaction** might create an image or sound, change a **variable**, or have many effects. Note any changes to **objects** or **variables** that will happen as a result of interactions.

CoderDojo

Produced by the CoderDojo Foundation // @CoderDojo

1 Surprise

A game should have **surprises** for the player, something new to discover or an **interaction** they haven't experienced in previous play-throughs. Without **surprise**, every challenge will be the same and the game will get boring fast.

Surprise comes from unpredictability. For example, a player who discovers an invisible block where he can jump and gain a bonus feels rewarded and satisfied. Try to hide little **secrets** in your game. Maybe there is a **secret** path to skip a difficult enemy or there is a strange **interaction** when two different enemies meet. Events like these are often called **"easter eggs"** because they are hidden and players hunt for them! Additionally, **surprise** also creates new challenges. If enemies appear in the same spot of the stage every time the player can learn to expect them, but if they appear in a random place the player will have to make a new decision every time.

If you want, add some **randomness** to your game. This means having the computer decide, at random, if or how something happens. **Randomness** adds to the game's replay-ability and helps make the player's experience more varied. It also ensures no two game sessions will be the same.

Update your plan

Get out your game plan again and update it with these answers:

- What will **surprise** the player?
- What **secrets** can the player find in your game?
- Can your game get better if you add some **randomness**? Where?

CoderDojo

Produced by the CoderDojo Foundation // @CoderDojo

2 Fun

A game should be **fun**. If the player is not having **fun** while playing they will quickly stop and never play again.

Different people enjoy different things. A game that one person likes might not be enjoyable to someone else. Some people prefer fast games with quick reactions, others prefer slower and more thoughtful games. Some people prefer immersive games where they forget the real world exists, others prefer a quick game between other activities. **You can't design for all of them**, and that's ok. Instead, make something that you find **fun** and you think others might find **fun** too.

Fun comes from challenge. If a game is too easy it will bore the player, if it is too difficult it will frustrate the player. A player will enjoy it when they know they came very close to **losing** but by being clever and quick, they managed to **win** instead.

Once you've made a basic version, in order to find out if your game is **fun**, you need to **playtest** it: Ask a friend to try the game. Try to tell them **nothing** about the game until after they've finished playing.

Update your plan

Get out your game plan again and update it with these answers:

- What makes the game **fun**?
- Is the game too easy or too difficult?
- Once it's ready: What do your **playtesters** tell you about the game?

CoderDojo

Produced by the CoderDojo Foundation // @CoderDojo

1 Theme

A game should ideally have a **theme**. Without it, players cannot connect emotionally and might have trouble remembering things. For example, chess is a game where you move black and white pieces. If pieces were numbered, it would be very difficult to remember how each piece moves. Instead you have two armies battling, pieces are named, and you can easily remember that the knight moves in an L shape and can jump with his horse over other units.

Decide on a **theme** that your game will focus on. A cool **theme** can help you decide on what a **character** looks like and what it can do, as well as what the environment and enemies will be. That also helps the player expect things, for example that a bird will fly or a wheel will roll. For example, a football game would look a bit different from a tennis game, but very different to a game where elephants play football. A cool **theme** can capture the player's imagination and help you make game decisions.

Update your plan

Get out your game plan again and update it with these questions and answers:

- What **theme** will the game have?
- What do the player's **character** and other **objects** look like? (Draw them if you like!)
- What will the player imagine being while playing the game?

CoderDojo

Produced by the CoderDojo Foundation // @CoderDojo

2 Story

Now that you've decided the game's **theme**, it's time to understand what will happen in the game's **story**. A game without a **story** might feel dull and pointless, without getting the player invested in playing. A **story** can transform a plain game about progressing through levels to an epic **story** about rescuing your eggs from evil pigs. Player's actions will have an effect on the things, saving the world or winning the gold medal.

Story also includes other **characters** that might be friendly and helpful or aggressive and disruptive. A **story** might have a good main character, a teammate and an evil villain. The **story** also takes place in a **game world**. It can be like our own, or it can be a completely new one of your imagination. You can decide that gravity does not exist, or that elephants eat clouds! A new and interesting **world** can make your game much more fun and engaging.

Update your plan

Get out your game plan again and update it with these questions and answers:

- What **world** is my game taking place in? What does it look like?
- What are the rules of that world? What is its geography? What kinds of creatures live there?
- What is the player's **story**? How does their game **goal** affect what their **character** is trying to do?
- What other **characters** are in the game? What do they do?   (Draw them if you like!)

CoderDojo

Produced by the CoderDojo Foundation // @CoderDojo

1 Start small

A great game is not one with a hundred different playable **characters** or a thousand levels, but one that has one playable **character** or five levels that are all very **fun**. A game that focuses too much on extra content without being **fun** in the first place can't keep the player's interest in order to experience all the content.

Before you plan on what you want to add to the game, first identify what your game **can't do without**. Find the smallest part of the game you can, build it and try it. If it's not good, you can try to fix the problems. If it is good, then you can start planning on adding more things.

Update your plan

Get out your game plan again and update it with this key answer:

- What is the game's **heart**? What does the game **need** and what is just **nice to have**?

CoderDojo

Produced by the CoderDojo Foundation // @CoderDojo

2 Putting everything together

A game has game **objects**, textures, backgrounds, animations, sounds, music, **controls**, **interactions**, rules, physics, menus, hidden **variables** and an ever-shifting game state. It might also have bugs, unintentional interactions, glitches and crashes!

It takes a lot of work to create all the required components then assemble them to make a playable game. The best way to do it is to break the whole game into smaller tasks and try to create the **smallest game you can** to start with.

A game is not done until it is **playtested** thoroughly. Ask your friends to play your game, ask your mentors to play your game, ask people from school to play your game! Hear them out, understand what they found fun, what worked well and what didn't, if the game was too short or too long, if they would play again.

Don't stop working on your game until you have had were 5 **playtests** where you decided to change nothing based on the feedback you got. When that happens: **congratulations, your first game is ready!!!**