

Detailed Game Specification: Geometry Dash

Course: COMP 2659, Winter 2026

Instructor: Nolan Shaw

Author(s): Robert Parker Hutcheson, Isaac Klein, Riley Gramlich,
Eduard Mykhailets

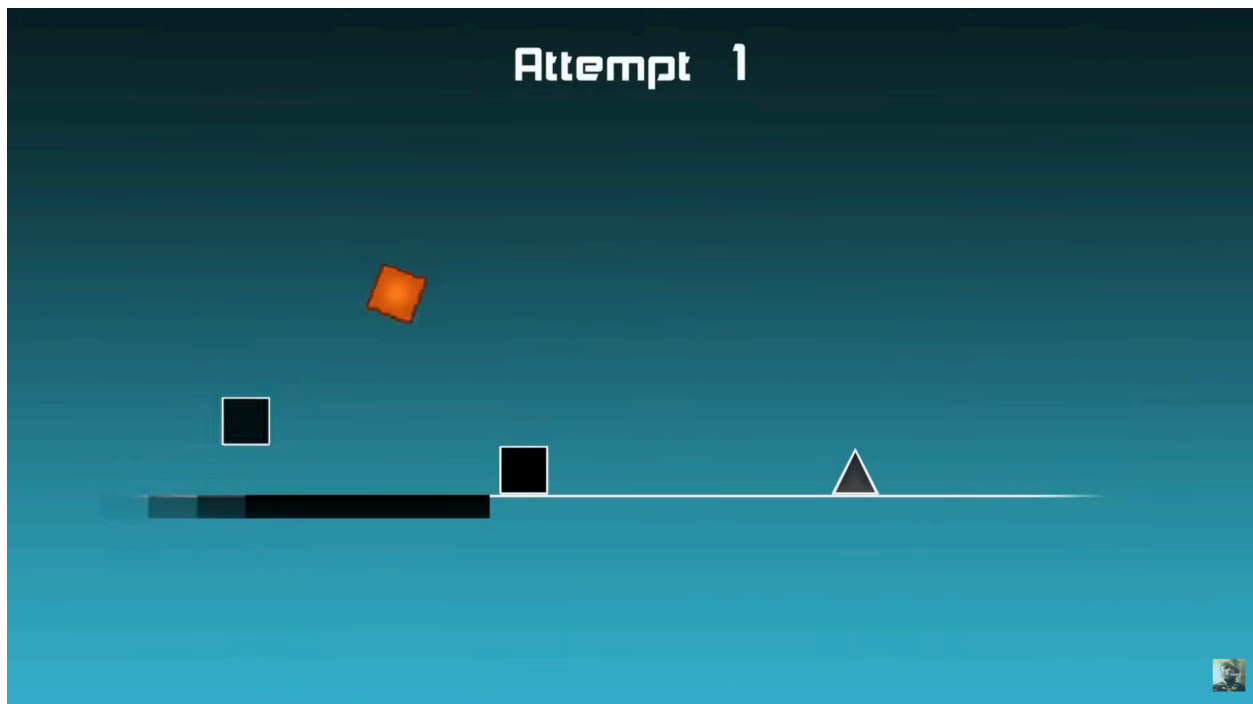
Last Modified: January 3, 2026

1. General Game Overview:

Geometry Dash is a single-player, platforming, side-scroller game that involves a square (named “Geo”) moving through an obstacle-filled level. The game is viewed from the side, and the entire screen represents a 2-D course that scrolls continuously to the right. The player controls Geo as he moves through the level automatically; the player does not control his speed or direction, only when he jumps. The objective of the game is to reach the end of the level without “dying”. Geo “dies” when he collides with a hazard: a spike, obstacle, or lava. At this point the attempt ends and the level restarts from the beginning.

Figure 1: Shows the square mid-jump as it clears a gap, with a spike ahead on the track that must be jumped over next.

Source: [1]



2. Game Play Details for Core 1-Player Version

Objectives and Rules:

A player is confronted with a static level consisting of obstacles. The level is traversed from left to right. The player is unable to control the rate at which they traverse the level; their horizontal velocity (dx) is held constant for the entire game. Player input is granted through the spacebar, which causes Geo to jump. Upon jumping, Geo will fall back down until encountering the ground or a platform. Geo will be unable to jump until he has landed onto a surface and is no longer in the air. The player must get Geo to jump throughout the course of the game to avoid obstacles until they either die or beat the level.

Obstacles consist of blocks, spikes, and lava. If spikes or lava are touched, Geo dies. If a block is landed on, that is the new floor until the Geo proceeds past the block. If Geo collides with a block from the left or bottom, i.e. he doesn't land on top, he dies. If Geo is not on a block, then the floor is the ground.

Once Geo has jumped through the static level, and made it to the end of the scrolling platform- you have beaten the level. Time permitting for level development, he will then proceed to the next level. Once the final level is completed, the player will be returned to the main menu.

If the player wins, there is no "score," they play until they win.

When the player dies, they restart the level.

When a player restarts the level, they start at the left-most position in the world coordinates ($player_x = 0$ and $player_y = 50$, subject to change).

Each level will not be detailed here, as it would be a series of hundreds of x/y values of obstacles. Furthermore, the physics will determine what sort of level design will lead to engaging gameplay. However, the levels are static.

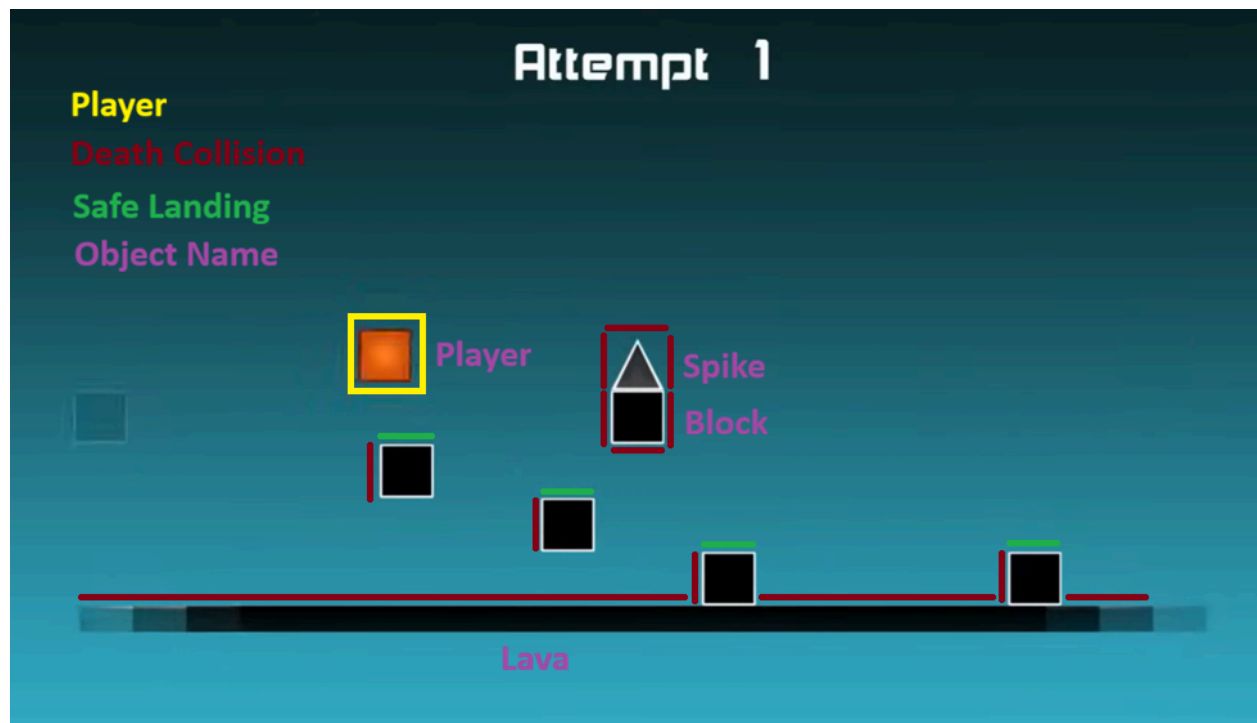
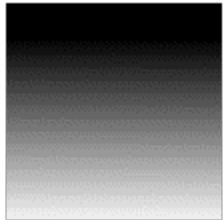
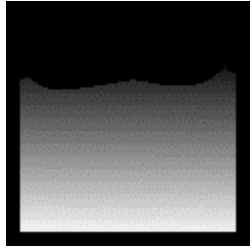
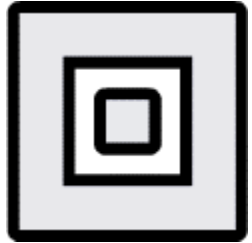


Figure 2: Annotated screenshot showing the objects present in The Impossible Game
Source: Adapted from [2]

Objects:

Object Name	Properties	Behaviours	Graphical Image
Block	(int) x: The x position of the top left of the object (int) y: The y position of the top left of the object (int []) const sprite*: The pointer for how to render this entity (int) const size: Width and height of this block	N/A	 Source: [3]

Spike	(int) x: The x position of the top left of the object (int) y: The y position of the top left of the object (int []) const sprite*: The pointer for how to render this entity (int) const size: Width and height of this spike Note: Spikes are treated as squares in shape at the start	N/A	 Source: [3]
Lava	(int) x: The x position of the starting left of the lava (int []) const sprite*: The pointer for how to render this entity (int) const size: Width of the lava	N/A	 Source: [3]
Geo	(int) x: The x position of the top left of the object (int) y: The y position of the top left of the object (int) dx: The change of the x position, or horizontal velocity (int) dy: The change of the y position, or vertical velocity (int) ddy: The change of dy, or vertical acceleration, or gravity (int []) const sprite*: The pointer for how to render this entity (int) const size: Width and height of the player	Move, constant movement towards the right (positive x) through the world. This occurs at a constant rate. Jump	 Source: [3]

Physics:

The game uses a simple, deterministic physics model focused on vertical movement and collision response.

Geo has a vertical position and vertical velocity. When the player presses the spacebar while on a valid surface, Geo is given a fixed upward velocity. Each frame, gravity is applied to Geo by reducing his vertical velocity by a constant amount. When the vertical velocity reaches zero, Geo transitions from upward motion to downward motion and continues falling until he collides with a surface.

Geo's vertical position is constrained by a minimum and maximum height. The minimum height corresponds to the ground or the current platform surface. The maximum height limits how far upward Geo can travel. If Geo's vertical position exceeds this maximum height during a jump, the visible play area is adjusted to keep him within the allowed vertical range, ensuring he remains visible on screen.

When Geo collides with the top surface of a block, he lands safely on top, his vertical velocity is set to zero, and the top of the block becomes the new effective ground level. Collisions with the sides of these blocks or with spike obstacles however immediately trigger the death condition.

Our world has “world coordinates” associated with it, and every obstacle/object in this world has a *fixed position* in this coordinate plane. As `player_x` increases, we bring objects into frame- effectively scrolling objects from right to left (note that Geo remains in a constant position on the screen- only Geo's x position *relative to the world* changes).

No rotational forces, acceleration-based horizontal movement, or friction are simulated. All physics calculations are performed using simple per-frame updates to ensure predictable behavior and consistent gameplay.

Synchronous Events:

Event Name	Trigger	Description
Move Obstacles	Every new frame, 1/70th of a second	We move objects “player_velocity” pixels to the left of the screen.
Move Geo	Every 1/70th of a second (every frame).	Geo is moved vertically by “player_dy” pixels, updating Geo's y position on the display. Geo does not move

		horizontally across the screen. Geo's "x value" represents his displacement in the world coordinate system, tracking the distance he has travelled through the world.
Player Gravity	Every 1/70th of a second (every frame).	Geo's y acceleration ("player_dy") should decrease and pull Geo to the ground or a platform, and should allow Geo to land/stay on platforms until the next jump

Asynchronous Events:

Event Name	Trigger	Description
Jump request	Space bar is depressed	This will set Geo's vertical velocity (dy) to 10. This will cause him to begin ascending vertically

Condition-Based (Cascaded) Events:

Event Name	Triggering Condition	Description
Collision	All objects will be kept to a width / height of 50 pixels, this means it is triggered when: $ \text{player_x} - \text{obstacle_x} < 50$ AND $ \text{player_y} - \text{obstacle_y} < 50$ Note: If we implement better collision for non-square objects, or change the size of objects, this formula will change.	This will trigger to check the sort of collision that occurs, generally speaking a death

Collision, sides or bottom	Collision (Cascaded) AND player_y <= obstacle_y	Player death (Geo dies)
Collision, top	Collision (Cascaded) AND player_y > obstacle_y	This will trigger to check if an obstacle is landable
Collision, landable	Collision, top (Cascaded) AND obstacle.isLandable()	A player will land on the top of the object, and set the floor height to player_x - 50, landed_object is set to this object.
Collision, not landable	Collision, top (Cascaded) AND !obstacle.isLandable()	Player death
Reset ground height	Collision, landable (Cascaded) will set the nearest_object, when player_x - obstacle_x > 50	The floor height is set to the ground height.
Landed	Floor height = player_y - 50	A player has landed on a floor, dy = 0
Not Landed	Floor height != player_y - 50	A player is not touching the floor.

Player death	Collision, sides or bottom (Cascaded) OR Collision, not landable (Cascaded)	The level is restarted
Level complete	Player_x = level_finish	The player has completed the level and is navigated back to the main menu / the next level is began depending on if it was the last level

Hypothetical Gaming Session:

The game begins with Geo positioned slightly away from the left edge of the screen, approximately quarter of the screen width from the left boundary, resting on the ground at the default floor height. Geo is stationary relative to the screen at the start of the game. The level consists of a predefined sequence of obstacles that approach Geo from the right side of the screen.

As the game starts, the player does not press any keys. Block obstacles begin to appear from the right and move toward Geo at a constant horizontal rate. The player remains at a fixed horizontal position on the screen while on the ground.

After a short time, a raised block approaches Geo. The player presses the space bar once, triggering a jump event. Geo gains an upward vertical velocity and rises to a height several times greater than the height of the block. While airborne, the Geo maintains his horizontal position on the screen as the block moves beneath it.

As gravity pulls Geo downward, he lands on the top surface of the block. Upon landing, the Geo's vertical velocity is set to zero and the top of the block becomes the new effective floor height. Geo continues forward as the block moves leftward and eventually exits the screen, at which point the square falls back down to the ground.

Shortly afterward, a spike appears on the ground ahead. The player delays the jump slightly longer than before and presses the space bar closer to the spike. This jump causes Geo to pass completely over the spike without making contact, landing safely on the ground on the other side.

Later in the level, the player mistimes a jump and collides with the side of an incoming block. This collision triggers a death event, immediately ending the current attempt. The level resets, and Geo is returned to his initial position, at the beginning of the level.

On the next attempt, the player successfully times each jump, landing on blocks when necessary and fully clearing spikes and gaps. Near the end of the level, a sequence of obstacles requires the player to jump onto a raised platform and then jump again to avoid a final hazard. Geo clears these obstacles and continues forward until reaching the level completion point.

When the player reaches the end of the level without dying, the level completion event is triggered. If additional levels are available, the game transitions to the next level. Otherwise, the player is returned to the main menu, indicating successful completion of the game.

3. Game Play Details for Core 2-Player Version

In the two-player version of Geometry Dash, both players race through the same side-scrolling level at the same time, each controlling a square with a single input (space/up-arrow) to jump. Player 2's Geo is shown slightly behind Player 1's: both players sharing the same world and obstacles. The player who survives the furthest wins the match. If one player dies, the remaining player can continue and finish the level.

4. Sound Effects:

The game uses a small number of sound effects to provide clear feedback for important game events without overwhelming the player.

Background music plays continuously during gameplay, different tracks for each different level. The music is rhythmically structured to complement the pacing of the level but does not provide explicit cues for when the player should jump. Gameplay remains fully visual, and successful timing is determined by obstacle positions rather than audio signals.

A short sound effect is played when Geo collides with an obstacle and the death condition is triggered. This sound provides immediate feedback that the current attempt has ended.

When the player successfully reaches the end of the level, a distinct completion sound is played to indicate success and reinforce level completion before transitioning to the next state of the game.

Death Sound sample: https://www.youtube.com/watch?v=XuPeF_YCsFE

Level Completion Sound: <https://www.youtube.com/watch?v=bfbFE4Ld848>

5. Additional Features (Time Permitting)

- Sound effects can be enabled or disabled independently of the background music.
- Additional obstacles/features such as big jump tile, midair jump, reverse gravity etc. As creative as complicated we got time to make.
- Synchronized music with the level design- matching the times when the player jumps to the beat of the music
- Attempts indicator, a counter with the number of failed attempts at the top of the screen
- Progress indicator, a status bar or percentage displayed at the top of the screen informing the player of their progress through the level (Example, 50% of the way to the end of the map).
- Checkpoints, specific locations that a player can restart from rather than starting from the beginning of the level
- File streaming- load assets and levels dynamically from files.

References:

- [1] AleXPain24. *"The Impossible game" - All 5 levels beaten (PC ver.)*. (May 15, 2014). Accessed: Jan. 18, 2026. [Online Video]. Available: https://www.youtube.com/watch?v=O_U65QkH_EU
- [2] AleXPain24. *"The Impossible game" - All 5 levels beaten (PC ver.)*. (May 15, 2014). Accessed: Jan. 18, 2026. [Online Video]. Available: https://www.youtube.com/watch?v=O_U65QkH_EU
- [3] Free_Games, *"Geometry Dash Assets,"* websim.com. [Online]. Accessed Jan. 23, 2026. Available: https://websim.com/@Free_Games/geometry-dash-image-viewer