Case 19

Nathan Fowler

Synchronized Hacking
Nathan Fowler

Revision History

VERSION	DATE	COMMENTS
1.0	11/17/10	Document created
1.1	11/22/10	System Redefined

Table of Contents

Game Overview:

Case 19 is a stealth action game that takes place in a small Ugandan village plagued by a terrible and mysterious disease. A journalist living in Uganda at the time asks for help from the United Nations, hinting that the origin of the disease is from a nearby nuclear power plant run by Asgard Industries. Jason Greer is sent in to speak with the journalist and investigate the source of the disease. Using his skills gained in the US Special Forces, Jason must uncover the truth and condemn those responsible by breaking into secure locations and confiscating incriminating evidence.

The co-op mode will be from the perspective of two different spies who are investigating different activities Asgard Industries is perpetrating elsewhere in the world. Ties to the single player game are crucial in order to bind the two stories together.

System Overview:

Hacking Synchronization

Hacking poses a higher challenge in the co-op mode than in single player with hacking synchronization. This system can be used to bring players together in order to bypass security systems such as locked doors. All hacks have a limited amount of time that they must be completed within.

In this system, players must perform several quick-time events in sync with each other in order to complete a hack. Both players begin with button prompts appearing on screen at the same time. But this changes as the hack progresses due to each individual's reaction time.

For example, if *Player A* hits his 1st prompt half a second before *Player B* hits her 1st prompt, then when the 2nd set of prompts appears, *Player A's* 2nd button prompt appears half a second before *Player B's* 2nd button prompt. This means that hacks can become out of sync if players do not communicate and time their button presses together. This de-synchronization of prompts continues to build until the end of the hack, and a player can time a late button press purposefully to help re-sync prompts.

Pressing the incorrect button during a prompt will not be valid, and a new prompt will appear within one second for that player to try again. If button presses are more than 3 seconds apart or if three mistakes are made collectively, the hack is failed.

One possible scenario with this system has players hacking two terminals on opposite sides of the room to open a door. A more complicated scenario would have players performing several synchronized hacks from several points in a guarded room.

Competitor Analysis:

Pros:

Creates interesting gameplay built around teamwork and communication outside of the game. Relatively simple to implement.

Cons:

New sound cues must be recorded and put into the system to give the status of hacks. Possibly difficult to communicate to players initially.

System Breakdown:

How:

Scenario 1

1. Players must be at their respective hacking terminals at the same time to initiate a synchronized hack.
2. While hacking, quick-time event buttons appear on each player's HUD (A, B, X, Y on Xbox for example). Both players must push the corresponding buttons and stay in sync in order to complete a hack. If a hack fails, there may be a consequence such as an alarm going off.

Components:

Art Team:

1. None (all assets already exist from single player)

Programming Team:

1. Making it so synchronized hacks cannot be started until both players are ready
2. Creating the time window for synchronized hacking events

Design Team:

1. Building system into levels

Narrative Team:

1. Dialogue for each character talking about starting a hack, current progress, time remaining, etc

Sound Team:

1. New feedback sounds for additional hacking mini-games

Possible Issues:

Communication between players online may be an issue if neither has a microphone. This system requires relatively precise communication, so taking away the ability to communicate may make this system frustrating to players. Lag while playing online could also render this system unplayable if it was bad enough since timing is so important. This must be compensated for in the programming.

References:

This is essentially cooperative quick-time events that must be performed in sync. Very few (if any) games have such a system already implemented, but there are many examples of quick-time events being used in single player games.

God of War 2

http://www.youtube.com/watch?v=ZaqoZlTNB3U