บทที่ 5
ระบบกล้องและการแสดงผลหน้าจอ

เกมมีกฎการมองเห็นของตัวเองซึ่งมักจะตรงกันข้ามกับกล้องประเภทอื่น ๆ การออกแบบกล้องจะกำหนดวิธีที่ผู้เล่นมองเห็นในโลกของเกมและท้ายที่สุดพวกเขาเล่นอย่างไรดังนั้นหากไม่มีการออกแบบกล้องที่ดีทั้งเกมของคุณก็อาจเล่นไม่ได้ และเป็นเรื่องยากมากที่จะเปลี่ยนกล้องที่ออกแบบมาไม่ดีโดยไม่ต้องเขียนเกมใหม่ทั้งหมด ภาพการแสดงผลของเกม (gamatography) (เหมือนการถ่ายภาพแต่สำหรับเกมแทน) ในการออกแบบหรือเขียนแม่แบบ (Prototype) ในทุกโครงการ น่าจะเป็นสิ่งแรก ๆ ที่จำเป็นต้องเขียนระบุ ส่วนเขียนคำสั่งที่เกี่ยวข้องกับกล้องก็จะเป็นสิ่งแรกด้วยเช่นเดียวกัน ดังนั้นจำเป็นต้องมีกฎเกณฑ์ที่ชัดเจน ซึ่งจะควบคุมกล้องตลอดทั้งเกม

เกมส่วนใหญ่ไม่ได้จำกัดขนาดของแผนที่ของตัวเองไว้ที่ขนาดหน้าจอเท่านั้น แต่จะสร้างแผนที่แทนใหญ่กว่าหน้าจอ เพื่อให้สามารถจัดการและแสดงแผนที่ดังกล่าวบนหน้าจอได้ ซึ่งการมองเห็นก็ต้องผ่านกล้อง บางครั้งเรียกว่า วิวพอร์ต (Vieport) ซึ่งจะช่วยให้สามารถแสดงเพียงส่วนหนึ่งของโลกหรือแผนที่ในเกมได้ นอกจากนี้ยังสามารถสร้างลูกเล่นต่าง ๆ ผ่านกล้องได้ซึ่งจะทำให้ผู้เล่นได้รับความบันเทิงหรือเข้าถึงเกมที่พัฒนาขึ้นมาได้มากขึ้นด้วย

โดยจะมี พื้นที่หน้าจอ (screen space) และพื้นที่เกม (game space) หรือบางครั้งสามารถเรียกว่า พื้นที่บนแผนที่ (map space) ซึ่งสามารถเขียนเป็นสมการได้ดังนี้

$$screen\ coordinates = map\ coordinates - viewport\ coordinates$$

$$map\ coordinates = screen\ coordinates + viewport\ coordinates$$

ประเภทกล้องที่ใช้มากที่สุด

เป็นรูปแบบการแสดงผลและการเปลี่ยนสิ่งต่าง ๆ ในการแสดงผลโดยผ่านกล้อง ซึ่งรูปแบบของกล้องที่นิยมใช้ในเกมมากที่สุด มีดังนี้

1. **กล้องคงที่ (Static Camera)**

เป็นรูปแบบที่ง่ายที่สุดที่สามารถนำไปใช้ได้ เป็นการแสดงผลหน้าจอเดิมโดยไม่มีการเคลื่อนไหวหรือเคลื่อนที่ ส่วนการเปลี่ยนหน้าจอสามารถใช้เทคนิคในการตัดภาพ (Cut) จางหาย (FateIn) หรืออื่น ๆ ได้เหมือนกับการทำคลิปวิดีโอ ตัวอย่างเช่น ทุกครั้งที่ออกจากแผนที่ หน้าจอจะจางหายไปเป็นสีดำและกลับไปที่ห้องที่ตัวละครอยู่ เป็นต้น

2. **กล้องกริด (Grid Camera)**
3. **กล้องติดตามตำแหน่ง (Position-Tracking Camera)**

เป็นกล้องที่ทำให้วิวพอร์ตจะติดตามตำแหน่งของตัวละครผู้เล่นและเคลื่อนที่ไปตามตัวละครไม่ว่าจะอยู่ส่วนในของแผนที่หรือฉาก ซึ่งที่ตัวละครอยู่ตรงกลางของหน้าจอ กล้องติดตามตำแหน่งที่ใช้ในวิดีโอเกมมี 2 ประเภท คือ กล้องติดตามแนวนอนและกล้องติดตามแบบเต็มรูปแบบ กล้องประเภทนี้อาจมีข้อเสียที่ร้ายแรงบางประการเมื่อเกี่ยวข้องกับการเปลี่ยนทิศทางอย่างกะทันหัน

หรือรวดเร็วมาก เนื่องจากกล้องติดตามตัวละครผู้เล่นตลอดเวลากล้องจะรู้สึกกระตุกและมีปฏิกิริยามากเกินไป ซึ่งอาจทำให้เกิดความไม่สบายใจหรือคลื่นไส้ได้

3.1 กล้องติดตามแนวนอน (Horizontal-Tracking Camera)

3.2 กล้องติดตามแบบเต็มรูปแบบ (Full-Tracking Camera)

4. กับดักกล้อง (Camera Trap)

ระบบ“ กับดักกล้อง” เป็นระบบถูกคิดค้นขึ้นเพื่อกำจัดหรือแก้ปัญหาที่ได้รับจากระบบของกล้องการติดตามตำแหน่ง

กล้อง. ตัวละครที่เล่นได้นั้นถูกห่อหุ้มด้วย "กับดัก" ซึ่งเมื่อ "Escape" ทำให้กล้องจับได้ด้วยความพยายาม

ทำให้ผู้เล่นกลับไปอยู่ใน "กับดัก" ดังกล่าว

กับดักนี้จะแสดงด้วยสี่เหลี่ยมผืนผ้าที่มองไม่เห็นในขณะเล่น แต่สามารถเปิดการมองเห็นได้บนหน้าจอในขณะทดสอบ ซึ่งวิธีนี้ช่วยให้กล้องกระตุกน้อยลงให้ความรู้สึกที่เป็นธรรมชาติมากขึ้น นอกจากนี้สามารถปรับขนาดกับดักกล้องได้ตามประเภทของเกมที่วางแผนไว้โดยการเขียนโค้ดเพิ่มเติม เกมที่เล่นช้าอาจมีกับดักกล้องที่ใหญ่กว่าทำให้สามารถใช้กล้องได้เพื่อพักบนหน้าจอเดียวกันมากขึ้น ในขณะที่เกมที่มีการเคลื่อนไหวเร็วขึ้นอาจมีกับดักกล้องขนาดเล็กเพื่อให้เวลาตอบสนองเร็วขึ้น

5. กล้องมองข้างหน้า (Look-Ahead Camera)

เป็นกล้องที่มีความซับซ้อนมากขึ้น ซึ่งจะใช้งานเมื่อตัวละครที่เล่นได้เคลื่อนที่ไปยังทิศทางที่แน่นอนอย่างรวดเร็ว. กล้องมองข้างหน้าใช้เพื่อแสดงพื้นที่ด้านหน้าของผู้เล่นมากขึ้นทำให้มีเวลามากขึ้นในการตอบสนองต่อสิ่งที่จะเกิดขึ้น เช่น อุปสรรคหรือศัตรู

กล้องนี้ต้องการการใช้งานที่ดีเมื่อต้องเปลี่ยนทิศทาง: มีการเปลี่ยนทิศทางอย่างกะทันหันตัวละครของผู้เล่นควรมีการตอบสนองอย่างช้าๆจากกล้องไปยังทิศทางใหม่ไม่เช่นนั้นจะทำให้ผู้เล่นรู้สึกคลื่นไส้ ดังนั้นกล้องนี้จึงไม่เหมาะอย่างยิ่งสำหรับเกมที่ต้องการการสร้างแพลตฟอร์มที่แม่นยำเนื่องจากจำเป็นต้องมีการแก้ไขการล็อกกล้องอย่างต่อเนื่อง

5. กล้องแนวทางไฮบริด

เป็นการประยุกต์หรือผสมผสานกล้องประเภทต่าง ๆ มาอยู่เกม เพื่อให้ความน่าสนใจหรือประสบการณ์ใหม่เพิ่มเติมให้ผู้เล่น และยังทำให้ดูเป็นเป็นเอกลักษณ์ของเกมได้อีกด้วย ตัวอย่างเช่นใน “Legend of Zelda: A link to the past” กล้องจะผสมผสานระหว่างไฟล์ "กับดักกล้อง" และ "กล้องกริด" ซึ่งแต่ละโซนเป็นส่วนหนึ่งของเส้นตารางและภายใน "เซลล์ตาราง" แต่ละอันเรามีการติดตามระบบขึ้นอยู่กับ "กับดักกล้อง" สิ่งนี้ช่วยให้เกมมีความรู้สึกไดนามิกมากขึ้น แต่ยังช่วยประหยัดหน่วยความจำเนื่องจาก SNES ต้องโหลด "โซน" เพียงโซนเดียวในแต่ละครั้งแทนที่จะเป็นแผนที่ทั้งหมด

https://translate.google.com/translate?sl=en&tl=th&u=https://www.whatgamesare.com/2011/10/camera-comes-first-game-design.html

กฎในการใช้กล้อง

Kelly, T. (2011 : 1-7)

1. กฎทอง (The Golden Rule)

การถ่ายภาพยนตร์เป็นเรื่องของการจัดองค์ประกอบของแสงและกล้องเพื่อถ่ายทอดเรื่องราวหรืออารมณ์ แม้ว่าหลาย ๆ เกมจะพยายามนำทั้งสองอย่างมารวมกัน แต่ความต้องการของกล้องเกมมักจะตรงข้ามกัน

ไม่มีใครอยากเห็นบั้นท้ายของเรือเดินทะเลอวกาศที่ปิดกั้นมุมมองของพวกเขา หรือการกระโดด 100 ครั้งเพื่อตัดการกระทำเพียงเพราะแอนิเมชันดูเท่ ๆ การเล่นเกมที่ดีต้องเริ่มจากสิ่งที่ผู้เล่นต้องการเห็นมากกว่าสิ่งที่ผู้พัฒนาต้องการบอกเล่า วิธีที่ผู้เล่นเห็นเกมและการตีความสิ่งที่เขาเห็นคือการมองผ่านเลนส์ (creative constant) ที่ส่งผลกระทบต่อทุกสิ่ง

กล้องถ่ายรูปเป็นส่วนสำคัญของค่าคงที่ดังนั้นกฎทองของการถ่ายภาพรังสีที่ดีคือ: หากกล้องของคุณไม่ช่วยให้ผู้เล่นเล่นได้ให้ออกแบบที่เหมาะสม

2. พื้นที่โดยรอบ (Surrounding Space)

เมื่อเล่นเกมมาริโอผู้เล่นไม่ได้มองไปที่มาริโอจริง ๆ เขากำลังดูแท่นแกว่ง เหรียญ ดาว เต่า และเห็ด ในเกมแข่งรถเขาไม่ค่อยชื่นชมรถของตัวเอง เขากำลังมองไปที่รถคันอื่นและลู่วิ่งคาดว่าจะถึงรอบต่อไป ในเกมกลยุทธ์เขาไม่ได้มองไปที่กองทหารของตัวเองเลย เขากำลังดูกองกำลังของฝ่ายตรงข้ามภูมิทัศน์ของการต่อสู้และทรัพยากรบนแผนที่ เกมที่เล่นตุ๊กตาผี ตุ๊กตาเป็นเพียงเครื่องมือในโลกและเป็นท่อส่งข้อมูลดังนั้นผู้เล่นจึงไม่ได้มองไปที่ตุ๊กตา เขามองไปที่พื้นที่โดยรอบตุ๊กตาแทน

โลกของเกมเป็นสิ่งที่ได้รับความสนใจจากผู้เล่นมากที่สุดเพราะเป็นที่ที่พันธมิตรและศัตรู สวิตช์ วัตถุปริศนา วิธีแก้ปัญหาที่เป็นไปได้นำเสนอตัวเอง พื้นที่โดยรอบของเกมแอคชั่นมักจะเป็นพื้นที่รอบ ๆ ตัวตุ๊กตาและมีกรวยยื่นออกมาจากด้านหน้า ในเกมกลยุทธ์พื้นที่โดยรอบอาจเป็นรูปโดนัทรอบ ๆ หน่วยของคุณหรือสนามรบระหว่างสองกองทัพ ใน FarmVille อาจเป็นพื้นที่เพาะปลูกของคุณมากกว่าชาวนาของคุณ

หากคุณพยายามบังคับให้ผู้เล่นมองไปที่ตุ๊กตาของพวกเขาเพื่อสร้างความสัมพันธ์ทางอารมณ์คุณก็แค่ปิดกั้นมุมมองของพวกเขาที่มีต่อพื้นที่โดยรอบ พวกเขามองเห็นด้านหลังของรถก้นของตุ๊กตาหรือมี 25% ของพื้นที่การมองเห็นถูกปิดกั้นด้วยไหล่ของตุ๊กตา อารมณ์เดียวที่แพร่พันธุ์คือความระคายเคือง

ดังนั้นกฎข้อที่สองของการออกแบบกล้องคือเพื่อให้แน่ใจว่ากล้องสามารถมองเห็นพื้นที่โดยรอบได้เสมอ เมื่อใดก็ตามที่การกระทำหลักของเกมมีแนวโน้มที่จะเกิดขึ้น (เทียบกับตุ๊กตา) จะต้องมองเห็นได้และหากนั่นหมายความว่าไอเดียระดับหรือแทร็กของคุณจะต้องถูกปรับรูปร่างใหม่เพื่อให้ทำงานร่วมกับกล้องได้นั่นคือสิ่งที่คุณต้องทำ

3. เอเจนซี่เท่าไหร่ (How Much Agency)

กล้องมักให้ระดับการควบคุมแก่ผู้เล่น มุมมองที่เคลื่อนไหวกลายเป็นส่วนหนึ่งของประสบการณ์การเล่นของเธอซึ่งทำให้โลกดูเหมือนสภาพแวดล้อมที่ลื่นไหลมากกว่าความท้าทายเชิงเส้น

อย่างไรก็ตามหากคุณไม่เคยเล่นเกมยิงมุมมองบุคคลที่หนึ่งโอกาสอาจเป็นเรื่องที่น่ากลัว ผู้เล่นต้องคุ้นเคยกับความคิดที่ว่าเธอสามารถควบคุมร่างกายส่วนบนและส่วนล่างของตนเองได้โดยอิสระจากกันและกัน ต้องใช้การฝึกฝนอย่างมากเพื่อให้สามารถทำการซ้อมรบเช่นการยิงกราดได้อย่างมั่นใจซึ่งเกมเมอร์ที่สนใจหลายคนได้มา แต่มือใหม่ยังขาด

ในทำนองเดียวกันประชากรส่วนใหญ่พบว่าเกมบางประเภททำให้สับสน นักเล่นเกมกลยุทธ์บางคนชอบกล้องที่สามารถเคลื่อนย้ายได้มากกว่ากล้องที่ลอยอยู่เพราะไม่เช่นนั้นพวกเขาจะหลงทางในแผนที่ บางคนรู้สึกได้โดยธรรมชาติแล้วว่าทางใดอยู่ทางเหนือและสามารถหาทางผ่านเมืองที่มีไบแซนไทน์ส่วนใหญ่ได้อย่างง่ายดาย คนอื่น ๆ หลงทางได้ง่ายแม้ในเมืองที่มีลักษณะคล้ายตารางเช่นชิคาโกโดยไม่มีจุดอ้างอิง

ผู้เล่นที่สับสนได้ง่ายจำเป็นต้องมีการต่อสายดิน เขาจำเป็นต้องรู้ว่าทางขึ้นหรือลงเหนือหรือใต้ซ้ายหรือขวาตลอดเวลา อย่างไรก็ตามสำหรับผู้เล่นที่สับสนทุกคนมีอีกคนหนึ่งที่พบว่าการขาดการควบคุมกล้องทำให้น่าหงุดหงิด เขาต้องการหมุนได้ 360 องศาในหลายแกน

ความตึงเครียดที่เกิดจากการขาดความเชี่ยวชาญหรือสับสนทั่วไปเมื่อเทียบกับความต้องการของผู้เล่นผู้เชี่ยวชาญนำไปสู่การปกครองที่สามของการออกแบบกล้อง: รู้ความอดทนของผู้ชมในการควบคุมกล้อง

อย่าพยายามโจมตีระบบของมนุษย์ต่างดาวให้กับผู้ชมที่ไม่มีความอดทนเพราะคุณจะทำให้พวกเขารู้สึกงี่เง่า แต่อย่าโง่ลงกล้องสำหรับผู้เล่นนินจาเพราะคุณจะทำให้พวกเขารู้สึกได้รับการอุปถัมภ์ พิจารณาว่าตลาดของคุณต้องการเพิ่มขีดความสามารถหรือถือครองไว้ด้วยมือจากนั้นออกแบบกล้องที่ตอบสนองความต้องการของพวกเขา

ไม่ว่าคุณจะทำอะไรอย่าพยายามทำทั้งสองอย่างพร้อมกัน วิธีนั้นคือความบ้าคลั่ง

4. ระยะทางในการดำเนินการ (Distance to Action)

แม้ว่าวิดีโอเกมอาจเป็นโลก 3 มิติ แต่ดวงตาของคุณทั้งคู่ก็มองเห็นภาพเดียวกันบนหน้าจอ ดังนั้นมันอาจดูเหมือนเป็นสเตอริโอ แต่จริงๆแล้วคุณกำลังดูเป็นโมโน คุณสามารถตัดสินระยะห่างระหว่างวัตถุที่อยู่ใกล้เท่า ๆ กันได้อย่างง่ายดาย แต่ไม่ใช่ว่าวัตถุเหล่านั้นกำลังเคลื่อนที่เข้าหาหรือห่างจากคุณ

เกม 2 มิติมักจะให้ความรู้สึกสะดวกสบายเพราะทั้งหมดทำงานบนระนาบเดียวกันเมื่อเทียบกับดวงตาของคุณ อย่างไรก็ตามสำหรับเกม 3 มิติไม่มากนัก การตัดสินการกระโดดใน Tomb Raider นั้นยากกว่า Sonic the Hedgehog มากเพราะการกระโดดนั้นเป็นแบบสามมิติ คุณไม่สามารถบอกได้ว่าหิ้งนั้นอยู่ไกลแค่ไหนสำหรับ Lara ดังนั้นคุณต้องกระโดดช้าๆและหวังว่าจะดีที่สุดในขณะที่โซนิคคุณสามารถกระโดดได้อย่างมั่นใจ 10 ครั้งภายในไม่กี่วินาที

ความลึกเป็นปัญหาน้อยกว่าที่ระยะทางซึ่งความแตกต่างระหว่างสเตอริโอและโมโนมีเล็กน้อย นักกีฬาคนแรกทำงานได้ดีจริงๆเพราะศัตรูมักจะอยู่ไกล ความแม่นยำเดียวที่ต้องการคือสองมิติอย่างมีประสิทธิภาพ (วางเรติคูลของคุณไว้ที่เป้าหมายแล้วยิง) อย่างไรก็ตามเมื่อศัตรูเคลื่อนเข้ามาใกล้ในเกมยิงมุมมองบุคคลที่หนึ่งผู้เล่นจะตระหนักอย่างถี่ถ้วนว่าเขามีข้อมูลเพียงเล็กน้อยเพียงใด

ดังนั้นกฎที่สี่ของการออกแบบกล้องคือการชดเชยสำหรับระยะทางที่จะดำเนินการ หากการกระทำหลักอยู่ไกลออกไป (เช่นเดียวกับเกมยิงมุมมองบุคคลที่หนึ่ง) จุดมุ่งหมายก็สำคัญกว่า หากเป็นภาพระยะใกล้ (เช่นเดียวกับในเกมต่อสู้) ความสามารถในการตัดสินระยะใกล้จะมีความสำคัญมากขึ้น หากคุณต้องการทราบข้อมูลมากมายจากทั่วทุกสนามรบความสามารถในการมองเห็นทั่วทั้งสนามก็มีความสำคัญ

อย่าพยายามบังคับให้มุมมองที่ใกล้หรือไกลออกไปเพียงเพราะมันดูเท่หรือดูเป็นภาพยนตร์ ผู้เล่นจะไม่สนใจว่าจะเล่นไม่ได้จริง

5. แสงสว่างนำทาง (Light The Way)

โดยปกติมนุษย์มีการมองเห็นในแนวนอน 170 องศาและแนวตั้ง 100 องศา แต่โชคดีที่มีหน้าจอ 100 องศาในแนวนอนและ 50 องศาในแนวตั้ง ในขณะที่เธอเล่นเกมของคุณผู้เล่นจะมองโลกที่ล้อมรอบด้วยขอบของหน้าจอเช่นไฟกะพริบของม้า

สำหรับกล้องถ่ายรูปส่วนใหญ่หมายความว่าผู้เล่นจะรับรู้โลกของเกมน้อยกว่าที่คุณคิด แม้ว่าเธอจะหมุนกล้องที่ลอยอยู่หรือหันศีรษะไปหาคนที่หนึ่ง แต่เธอก็ยังคงเห็นโลกในหน้าต่างที่อยู่ตรงหน้าเธอ ดังนั้นจึงง่ายมากที่จะทำให้เธอตาบอด

การมองเห็นไม่ชัดมีผลต่อการออกแบบเกมมากมาย ส่วนใหญ่ก็หมายความว่ากล้องและแสงความต้องการที่จะนำไปสู่การติดตามมากกว่า นั่นคือกฎข้อที่ 5 ของ gamatography หมายความว่าอย่างไร?

เทคนิคที่พบบ่อยที่สุดอย่างหนึ่งในโรงภาพยนตร์คือเมื่อนักแสดงปล่อยให้กล้องดันตัวเอง ในขณะที่เขาเดินจากซ้ายไปขวากล้องจะเริ่มติดตามเขาก็ต่อเมื่อเขาไปถึงด้านขวามือที่สามของหน้าจอ ซึ่งดูเหมือนว่าจะสามารถผลักตัวเองได้

ในเกมมักจะตรงกันข้าม ขณะที่ฉันดันจานบินตัวน้อยของฉันไปทางขวาใน Insanely Twisted Shadow Planet กล้องจะเคลื่อนที่ทันทีและเร็วกว่าที่ฉันทำผลักเรือลำเล็กของฉันไปทางด้านซ้ายของหน้าจออย่างมีประสิทธิภาพและให้แสงสว่างแก่เส้นทางข้างหน้า เกมนี้ยังจัดโครงสร้างเนื้อหาใหม่ส่วนใหญ่ไปในทิศทางของกล้องใหม่ดังนั้นฉันจึงไม่ค่อยถูกโจมตีจากด้านหลัง การโจมตีฉันจากทิศทางการเคลื่อนไหวให้ความรู้สึกยุติธรรมเพราะฉันเห็นว่ามันกำลังมา การโจมตีจากด้านหลัง (ที่ฉันทำไม่ได้) ไม่ทำ

กล้องถ่ายภาพบุคคลที่ลอยน้ำได้และกล้องถ่ายภาพบุคคลที่หนึ่งล้วนมีจุดอ่อนในเรื่องของการจัดแสง แต่เกมที่ใช้แสงจากกล้องสัญญาณหรือไฟแสดงสถานะภายในพื้นที่โดยรอบเพื่อกระตุ้นความสนใจของผู้เล่น ตัวอย่างเช่น Valve เป็นผู้เชี่ยวชาญในการใช้แสงเพื่อบอกใบ้ผู้เล่นอย่างต่อเนื่องโปสเตอร์เท่ ๆ บนผนังป้ายบอกทางออกไฟกะพริบที่น่าสงสัยและอื่น ๆ นำผู้เล่นไปตามเส้นทางที่ตั้งใจไว้โดยไม่ต้องควบคุมห่างจากเขาเลย

เกมอื่น ๆ ใช้ลูกศรหรือเครื่องหมาย Halo ใช้รูปแบบของตัวละคร Cortana นำทางหัวหน้าหัวหน้าเพื่อทำเครื่องหมายเป้าหมายบน HUD ของเขาและรักษาการออกแบบระดับผู้เล่นเดี่ยวให้เป็นเส้นตรง Grand Theft Auto ใช้ระบบลูกศรบ่งชี้และจุดที่ซ้อนทับโลก 3 มิติของเมืองเพื่อบอกผู้เล่นว่าจะไปที่ไหน

นอกจากนี้กล้องที่ถอดออกได้มักถูก จำกัด ด้วยกำแพงหรือสิ่งกีดขวางอื่น ๆ ซึ่งอาจทำให้การนำกล้องเป็นเรื่องยาก ดังนั้นกำแพงเหล่านั้นจำเป็นต้องถูกลบออกหรือขยายออกไป มิฉะนั้นจะนำไปสู่พฤติกรรมเช่นผู้เล่นขยับตุ๊กตาไปยังตำแหน่งที่ไม่เหมาะสมเพียงเพื่อให้กล้องหมุน หากเกมของคุณเป็นการผจญภัยแบบบุคคลที่สามที่มีทางเดินแคบให้ใช้กล้องติดตามแทน

6. การเปลี่ยน (Transitions)

เกมสมัยใหม่มักใช้กล้องมากกว่าหนึ่งประเภท Halo ใช้สองคน: คนแรกเมื่อคุณเป็น Master Chief แต่สามารถเคลื่อนย้ายได้เมื่อคุณอยู่ในยานพาหนะ Ico ส่วนใหญ่ใช้กล้องติดตาม (และผู้เล่นมีการควบคุมการหมุนที่ จำกัด) แต่ใช้กล้องเลื่อนที่ส่วนผนัง

การใช้กล้องหลายตัวด้วยวิธีนี้ไม่มีอะไรผิด ส่วนที่ซับซ้อนคือการจัดการการเปลี่ยน โรงภาพยนตร์มักใช้การตัดอย่างไรก็ตามเกมมักจะตอบสนองได้ดีกว่าด้วยการเคลื่อนไหวมากกว่าการตัดเนื่องจากการตัดระหว่างการเล่นทำให้สับสน

Halo ใช้ลำดับแอนิเมชั่นความยาวหนึ่งวินาทีของ Master Chief ในการเข้าหรือออกจากยานพาหนะเพื่อเป็นโอกาสในการเปลี่ยนแปลง กล้องจะเคลื่อนออกจากด้านหลังศีรษะไปยังจุดชมวิวด้านหลังรถและการถอยหลังจะเกิดขึ้นเมื่อเขาออกไป วินาทีนั้นเพียงพอสำหรับผู้เล่นที่จะเข้าใจการเปลี่ยนโหมด ในที่สุดเขาก็แทบไม่สังเกตเห็นมัน สมบูรณ์แบบ.

บางครั้งการเปลี่ยนการเคลื่อนไหวจะไม่ทำงาน เมื่อผ่านประตูในเกมแอ็คชั่นบุคคลที่สามมักจะไม่สามารถทำได้ที่จะเลื่อนกล้องติดตามไปที่ใต้ประตูและสำรองข้อมูลอีกครั้ง เกมจะต้องหยุดชั่วคราวนานกว่าหนึ่งวินาทีหรือเคลื่อนกล้องผ่านกำแพง ไม่ดีเท่าไหร่กล้องก็เลยตัดแทน มันอาจจะเลือนหายไปในชั่ววินาทีเช่นเดียวกับใน Ico อย่างไรก็ตามการตัดสีซีดจางนั้นไม่เหมาะเสมอไป

วิธีหนึ่งในการลดความสับสนจากการตัดที่เร็วขึ้นคือการรักษาทิศทางการเคลื่อนไหวจากกล้องตัวก่อนหน้า นอกจากนี้ยังควรกำหนดกฎที่จะไม่มีการดำเนินการใด ๆ เกิดขึ้นทันทีที่อีกด้านหนึ่งของทางเข้าประตู มันไม่ยุติธรรมสำหรับผู้เล่นที่จะคาดหวังให้เขาปรับทิศทางและปรับตัวให้เข้ากับความท้าทายทั้งหมดในวินาทีเดียวกัน ให้เวลาเขาสักครู่เพื่อพิจารณาว่าทางขึ้นหรือลงก่อนที่จะโจมตี

โดยรวมแล้วกฎข้อที่ 6 ของการออกแบบกล้องคือ: วางพื้นฐานการเปลี่ยนภาพของคุณไว้เสมอให้เวลาผู้เล่นสักสองหรือสองวินาทีเพื่อรับตำแหน่งของเขาก่อนที่จะไปทำภารกิจต่อไป เป้าหมายของการเล่นเกมที่ดีคือเพื่อให้กล้องรู้สึกเป็นธรรมชาติจนเป็นส่วนหนึ่งของสิ่งที่เกมนี้เป็นมากกว่าระบบที่ต้องต่อสู้ การจัดการให้ดีตั้งแต่เนิ่นๆจะแจ้งให้ทราบถึงการตัดสินใจหลายอย่างเช่นโมเดลระดับรายละเอียดที่ต้องมีและจำเป็นต้องเป็น 3D อย่างสมบูรณ์หรือไม่ นอกจากนี้ยังมีส่วนสำคัญในการออกแบบระดับ

ตามหลักการแล้วคุณควรไปถึงจุดที่กล้องเกมมีความแข็งแกร่งและทำงานได้อย่างสม่ำเสมอตลอดทั้งเกม แทนที่จะเป็นเรื่องน่าตื่นเต้นกล้องเกมควรจะคาดเดาได้แม้จะน่าเบื่อ เมื่อคุณมีกล้องถ่ายรูปก็เป็นเพื่อนของคุณมากกว่าศัตรูและสิ่งที่ยิ่งใหญ่ก็จะปรากฏออกมา

มีข้อยกเว้นสำหรับกฎทั้งหมด แต่หกข้อข้างต้นเป็นการเริ่มต้นที่ดี หากคุณเห็นภาพได้ว่าต้องการใช้กล้องประเภทใดและจะจัดการปัญหาต่างๆเช่นการเปลี่ยนภาพหรือพื้นที่โดยรอบอย่างไรก่อนที่คุณจะสร้างเกมจำนวนมากก็จะต้องใช้เวลาอย่างคุ้มค่า

เหนือสิ่งอื่นใดไม่ได้รับการติดอยู่ในความคิดว่างานของกล้องที่จะทำให้ผู้เล่นรู้สึก นั่นอาจเป็นความจริงในสื่ออื่น ๆ แต่ไม่ใช่ในเกม หน้าที่ของ Gamatography คือให้ผู้เล่นเล่นและสร้างการเชื่อมต่อทางอารมณ์ด้วยตัวเอง ดังนั้นให้ผู้กำกับภาพยนตร์กำลังภายในของคุณออกไปและมุ่งเน้นไปที่สิ่งที่ผู้เล่นต้องการก่อน

Kelly, T. (2011). Camera Comes First [Game Design].
https://www.whatgamesare.com/2011/10/camera-comes-first-game-design.html

มุมมองของกล้อง

เป็นหน้าจอเหมือนกับกล้องที่จะแสดงผลให้เราเห็นในฉาก ๆ นั้น บางครั้งอาจ กล้องอาจติดตามตัวละครผู้เล่นไปยังที่ต่าง ๆ ในฉาก หลายคนอาจจะเรียกว่า Player Layer ทั้งนี้ก็ขึ้นอยู่กับรูปแบบของกล้องด้วย

First Person : FPS
Third Person : TPS
Three-quarters View
2.5-D View
Forced Scroll
Spline
Locked Camera

การออกแบบองค์ประกอบของเลเยอร์

ซึ่งจะเกี่ยวการออกแบบเลเยอร์นั้น จะประกอบไปด้วย

1. **HUD Layer**

เป็นหน้าจอที่แสดงข้อมูลที่อยู่บนหน้าจอของผู้เล่น

พลังชีวิต/สถานะ / ชีวิตที่ยังสามารถเล่นต่อได้
เงิน / คะแนน / พลังงาน / เชื้อเพลิง /
ความสามารถ / ทักษะ / ระดับ (Rank)
เวลาที่ใช้ไป/เวลาที่เหลืออยู่ (Count Down)
แผนที่ / ระบบนำทาง (ที่สามารถเชื่อมต่อนอกฉากได้)

ความเร็ว

2. **Popup Layer**

เป็นหน้าจอสำหรับแสดงขึ้นในฉาก เพื่อวัตถุที่ต่างกัน

หยุดการเล่น

เลือกไอเท็ม

ดูแผนที่ / ดูข้อมูล

การดาวน์โหลด

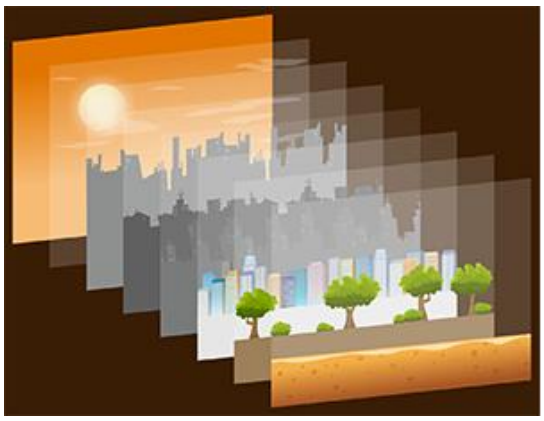
การขึ้นข้อความ ของระบบ / NPC / Monster

การได้คะแนน / บรรลุเป้าหมาย

3.

4. **Parallax Layer**

เป็นเลเยอร์ที่วางซ้อนกัน โดยมีการเปลี่ยนแปลงที่แตกต่างกัน ผ่านการเคลื่อนไหวของกล้อง ซึ่งจะเหมาะกับเกมแนว Side-Scrolls



Cutscene

7.7.1 Videos

วิธีที่ง่ายที่สุดในการใช้คัตซีนในเอ็นจิ้นและเฟรมเวิร์กส่วนใหญ่คือการใช้วิดีโอ และเฟรมเวิร์กส่วนใหญ่มีรองรับการสร้างหรือใช้ไฟล์มัลติมีเดียโดยใช้คำสั่งไม่กี่บรรทัด

7.7.2 Scripted Cutscenes

Loading Screens

การโหลดหน้าจออาจขึ้นอยู่กับการออกแบบเช่นกันและการตัดสินใจว่าจะใส่อะไรลงไปจะช่วยเพิ่มประสบการณ์ของผู้เล่นกับผลิตภัณฑ์ของคุณได้

1. สิ่งที่สามารถในฉากโหลดเกม

กล้อง

Effect Layer

จะเป็นเลเยอร์ ใช้แสดงในส่วนที่เป็น Effect บางอย่าง เพื่อให้รู้สึกสมจริงมากขึ้น เช่น หน้าจอถูกการโจมตี หมอกหนา ที่มืด

1. blood Screen / Damage Screen
2. Speed line
3. Dark Area
4. Time Scale
5. Screen Shake

เมื่อใดก็ตามที่เราต้องการแสดงแผ่นดินไหว การระเบิด หรือเอฟเฟกต์อื่น ๆ ในเกมของเรา เอฟเฟกต์การสั่นไหวของกล้องก็มีประโยชน์

(https://www.toptal.com/unity-unity3d/2d-camera-in-unity)

Lane System

ตอนนี้กล้องจะติดตามผู้เล่นในแนวนอนเท่านั้นเราจึง จำกัด ไว้ที่ความสูงของหน้าจอเดียว ถ้าตัวละครปีนบันไดหรือกระโดดสูงกว่านี้เราต้องทำตาม วิธีที่เราทำคือการใช้ระบบเลน

เริ่มแรกตัวละครจะอยู่เลนล่าง ในขณะที่ตัวละครยังคงอยู่ในขอบเขตของเลนนี้กล้องจะเคลื่อนที่ในแนวนอนเท่านั้นโดยชดเชยความสูงเฉพาะเลนที่เรากำหนดได้

ทันทีที่ตัวละครเข้าสู่เลนอื่นกล้องจะเปลี่ยนไปยังเลนนั้นและเคลื่อนที่ในแนวนอนจากจุดนั้นไปเรื่อย ๆ จนกว่าจะมีการเปลี่ยนเลนถัดไป

ต้องใช้ความระมัดระวังในการออกแบบเลนเพื่อป้องกันการเปลี่ยนเลนอย่างรวดเร็วในระหว่างการกระทำเช่นการกระโดดซึ่งอาจสร้างความสับสนให้กับผู้เล่น ควรเปลี่ยนเลนก็ต่อเมื่อตัวละครของผู้เล่นอยู่ในนั้นสักพัก

ระดับของเลนสามารถเปลี่ยนแปลงได้ตลอดทั้งระดับเกมตามความต้องการเฉพาะของนักออกแบบหรืออาจถูกขัดจังหวะทั้งหมดและระบบติดตามกล้องอื่นสามารถเข้ามาแทนที่ได้ ดังนั้นเราจึงต้องมีลิมิต จำกัด สำหรับการระบุโซนเลน

ระบบกล้อง

1. **ระยะการจับกล้อง Side-Scrollers แบบ Camera-window**

   Camera-window เป็นการดันตำแหน่งกล้องเมื่อผู้เล่นชนขอบหน้าต่างของกล้อง

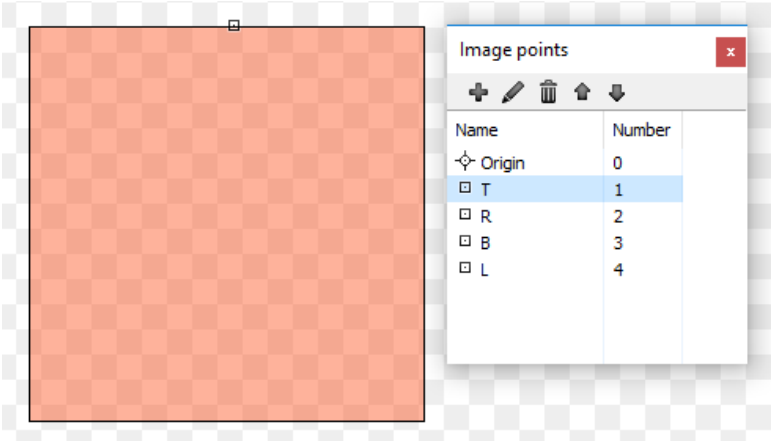
Jump Bug © 1981 Hoei/Coreland (Alpha Denshi)

Rastan Saga © 1987 Taito

ที่มา : Itay Keren. (2015). The Theory and Practice of Cameras in Side-Scrollers. Retrieved 15 July 2019, form https://docs.google.com/document/

ขั้นตอนการกำหนดคำสั่งต่าง ๆ ใน Construct 2

1. สร้าง sprite สี่เหลี่ยมจัตุรัส 250*250
2. เทสีตามที่ต้องการ โดยให้ใช้ Alpha <=100
3. ตั้งค่า Image point เพิ่มอีก 4 จุด โดยที่ T = กึ่งกลางขิดบน R = กึ่งกลางขิดขวา B = กึ่งกลางขิดล่าง L = กึ่งกลางขิดซ้าย ซึ่งจะกำหนดตามรูป



4. ใส่พฤติกรรม (Behavior) เป็น Scroll To ให้กับ Camera
5. ตัวละคร (Player) ตั้งชื่อตามความเหมาะสม ซึ่งในตัวอย่างนี้จะใช้ชื่อว่า น้องหมา และเพิ่มตำแหน่งของ Image point ในตำแหน่งหน้าสุด และบนสุด โดยในภาพนี้จะไว้ที่จะปากหมา ซึ่งก็สามารถทำได้เช่นกัน (เป็นตัวตรวจสอบระยะของกล้อง)

6. จากนั้นกำหนดค่าดังภาพต่อไปนี้

กล้องขวา			
System	น้องหมา.ImagePointX(1) > camera.ImagePointX(2)	camera	Set X to *camera.X+int(น้องหมา.ImagePointX(1)-camera.ImagePointX(2))*
		Add action	
กล้องซ้าย			
System	น้องหมา.ImagePointX(1) < camera.ImagePointX(4)	camera	Set X to *camera.X+int(น้องหมา.ImagePointX(1)-camera.ImagePointX(4))*
		Add action	
กล้องบน			
System	น้องหมา.ImagePointY(1) < camera.ImagePointY(1)	camera	Set Y to *camera.Y+int(น้องหมา.ImagePointY(1)-camera.ImagePointY(1))*
		Add action	
กล้องล่าง			
System	น้องหมา.ImagePointY(1) > camera.ImagePointY(3)	camera	Set Y to *camera.Y+int(น้องหมา.ImagePointY(1)-camera.ImagePointY(3))*
		Add action	

หมายเหตุ ตัวเลขที่อยู่ใน ImagePointY(1) คือ ตัวเลขที่กำกับอยู่ใน Image point ของแต่ตัว

หมายเหตุ ถ้าต้องการให้กล้องวิ่งตาม (เหมือนกล้องวิ่งตามไม่ทัน) ใช้ lerp(a,b,x) โดยที่ a ตำแหน่งเริ่มต้น, b ตำแหน่งปลายทาง, x ความหน่วง (1-0.01 ซึ่งยิ่งค่าน้อยจะหน่วงมาก และ 1 จะไม่หน่วงเลย แต่ถ้าจะเอา ตั้งแต่ 0.5-1.0 แทบจะไม่รู้สึกถึงความแตกต่างเลย ดังนั้นถ้าให้ดีก็ไม่ต้องใช้เพราะประมวลผลได้เร็วขึ้น

```
lerp(Self.X,camera.X+int(น้องหมา.ImagePointX(1)-camera.ImagePointX(2)),0.3)
```

2. **Smooth Camera**

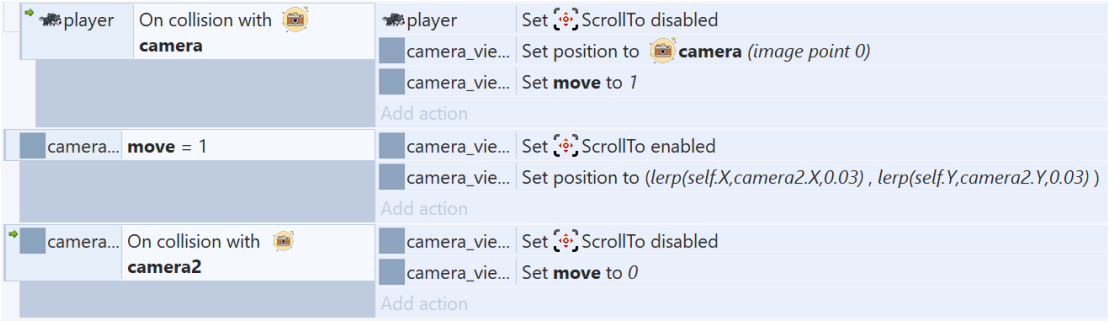
   จะเป็นการสร้างกล้องที่จับตามวัตถุ ซึ่งจะเป็นเหมือนกล้องจริงๆ

   2.1 สร้างวัตถุเพื่อใช้แทนเป็นกล้อง [Behavior : Scroll To]

   2.2 เขียนคำสั่งให้วัตถุตามตัว Player ดังนี้

3. **การเปลี่ยนกล้องไปยังอีกหน้าจออื่น**

   https://www.youtube.com/watch?v=9H4wDKvW2X0



จากคำสั่งจะเป็นการที่ตัว Player ไปชนกับไอคอน Camera ซึ่งทำให้

1. คำสั่ง ScrollTo ที่ติดอยู่กับ Player ปิดการทำงาน
2. ให้วัตถุ camera_view ไปที่ตำแหน่ง ไอคอน Camera
3. ตั้งค่า move = 1 ซึ่งเป็นตัวแปรใน camera_view
4. เมื่อ move = 1 คำสั่ง ScrollTo ที่ติดอยู่กับ camera_view จะเปิดการทำงาน
5. จากนั้น วัตถุ camera_view จะเคลื่อนไปยัง ไอคอน Camera2
6. เมื่อวัตถุ camera_view เคลื่อนไปถึงไอคอน Camera2 คำสั่ง ScrollTo ที่ติดอยู่กับ camera_view ปิดการทำงาน
7. ตั้งค่า move = 0 เพื่อให้ยกเลิกการเคลื่อนที่

lerp(self.X,camera2.X,0.03)

เป็นคำสั่งให้เคลื่อนที่จากตำแหน่ง A ไปยัง ตำแหน่ง B โดยจำเป็นต้องใส่จำนวนหน่วงของเวลาจากภาพ ใช้ 0.03 วินาที

4. การซูม