

זיהוי התנגשות

ל-OpenGL אין מידע על האובייקטים שהוא מצייר, והוא לא מסוגל לזהות הנתגשות ביניהן. אבל במשחקים/סימולציות אנחנו נרצה לזהות מתי שני אובייקטים מתנגשים, ולהחליט מה לעשות במקרה כזה - בד"כ באמצעות כללים פיסיקליים פשוטים. הנחת ההעבודה שלנו היא שכל האובייקטים הם פוליגונים תלת מימדיים (למרות שלא חייבים להניח את זה)

ההתנגשות מתרחשת בתאווה - ולכן נבדוק אותה בתוך האירוע שאחראי על התאווה:

1. idle function

2. קוראים ל-animation function

3. בודקים אם הייתה התנגשות

4. אם הייתה - מזיזים את האובייקטים בהתאם

5. מציירים

אנחנו רוצים זיהוי מהיר, ולכן נשתמש במודל שאינו דייקני על המילימטר כדי לקבל תוצאות מספיק טובות בזמן חישוב מהיר.

התנגשות בין נקודה לפוליגון

למשל - אם ממדלים את השחקן בתור נקודה, ורוצים לדעת אם הוא נתקל בקיר. נסתכל על נקודה בתוך הפוליגון, ועל הקוים המחברים אותה לכל אחת מקודקודי הפוליגון. סכום הזוויות בין הקוים יהיה 360° . אבל אם היא תהיה בחוץ - הוא יהיה שונה מ- 360° . ואם היא תזוז לאורך הנורמל, הסכום יהיה קצת קטן יותר מ- 360°

- בעיות:
- דיסקרטיזציה - התאווה היא בקפיצות. יכול להיות שבפריים אחד נהיה מצד אחד של הפוליגון ובפריים אחר נהיה מהצד השני שלו. לכן נרצה אולי להשתמש בסף (למשל $\sum \alpha \geq 350^\circ$)
 - גודל הפאה מאוד חשוב - אם הפאה קטנה אז השינוי בזווית הוא מאוד מאוד חד. לכן הסף צריך להיות תלוי בגודל הפוליגון.
 - אם יש גוף שמורכב מהמון פוליגונים להתחיל לחשב עבור כל פוליגון ופוליגון יהיה.

השיטה הזאת טובה למשטחים גדולים ואחידים כמו קירות, אבל לא למשטחים קטנים.

התנגשות Polygon Mesh

אנחנו צריכים למצוא דרך למדל את האובייקט שלנו בצורה כזאת שיהיה קל ומהיר לבצע חישובים.

Bounding Sphere

השיטה הכי מהירה היא לעטוף את האובייקט בכדור, ולבדוק אם הכדורים שעוטפים את האובייקטים מתנגשים. מגדירים את מרכז האובייקט בתור הממוצע בין כל הקודקודים, והמרחק לקודקוד הכי רחוק הוא הרדיוס. כדי לבדוק אם יש התנגשות, פשוט בודקים:

- בשביל לבדוק התנגשות בין נקודה לכדור, מחשבים את המרחק מהנקודה למרכז הכדור $D^2 = \Delta X^2 + \Delta Y^2 + \Delta Z^2$ ובודקים אם $D^2 < R^2$.

- אם רוצים לבדוק התנגשות בין שני כדורים, מוצאים את המרחק בין המרכזים שלהם ובודקים אם $D^2 < (R_1 + R_2)^2$

- אם רוצים לבדוק התנגשות בין כדור למישור, צריך לחשב את המרחק מהכדור למישור. אם יש לנו נקודה P על המישור והנורמל שלו זה N , ומרכז הכדור הוא C , אז $D = (C - P) \cdot N$.
– הנורמל מציין את הכיוון החוצה - אם נעבור את הקיר נקבל $D < 0$.
אפשר ליעל את החישוב אם נחסוך את חיבור הווקטורים ונהפוך את זה לחיבור סקלרים: $D = C \cdot N - P \cdot N$.

Bounding Cylinder

אם יש לנו אובייקטים צרים וארוכים (למשל עמודים), נרצה לתחום אותם עם צילינדרים. אם הצילינדר עומד, אז המעגל הוא על מישור ה XZ והגובה הוא על ציר ה Y . המרכז יהיה ממוצע הקודקודים, הרדיוס יהיה המרחק על מישור ה XZ לקודקוד הרחוק ביותר, ואז צריך למצוא את ה Y המינימלי והמקסימלי. כדי לבדוק התנגשות בודקים את המרחק במישור ה XZ בין הנקודה במרכז הצילינדר לנקודה שאנחנו רוצים לבדוק. אם הוא גדול מהרדיוס, אין התנגשות. אם הוא גדול - צריך לבדוק שהנקודה נמצאת גם בתוך תחום ה Y .

Bounding Box

אפשרות נוספת היא לתחום באמצעות קופסא. הקופסא הכי פשוטה היא Axis Aligned Bounding Box - קופסא בהתאם לצירים, ואז החישובים יותר פשוטים. כדי למצוא את הקופסא, מוצאים את המינימום והמקסימום על כל אחד מהצירים. כדי לבדוק התנגשות, נבדוק כל אחד מהצירים בנפרד. כדי לבדוק התנגשות בין קופסא לכדור, נחשב תחילה את המרחק בין המרכזים: $D^2 = \Delta X^2 + \Delta Y^2 + \Delta Z^2$. אם $D^2 < R^2$ (כש R זה רדיוס הכדור) אז יש התנגשות. אחרת, צריך להתחיל לבדוק:

```
for (int i = 0; i < 3; ++i) {  
    if (C[i] < B.min(i)) {  
        s = C[i] - B.min(i);  
        d += s * s;  
    } else if (C[i] > B.max(i)) {  
        s = C[i] - B.max(i);  
        d += s * s;  
    }  
}
```

כלומר בודקים עבור כל אחד מהצירים אם מרכז הכדור נמצא בתוך הקופסא בהתחשב באותו ציר, ואם לא אז מוסיפים את המרחק מהדופן הרלוונטית על אותו ציר לחישוב המרחק.

Oriented Bounding Box

לפעמים קופסא מיושרת לצירים לא מספיקה, ואנחנו רוצים קופסא שנמצאת בזווית יותר מתאימה. יש שתי שיטות: להגדיר אותה יחד עם האובייקט, או לחשב אותה. כדי לחשב אותה:

- מוצאים תחילה את המרכז בתור ממוצע הקודקודים.
- מחפשים את הקודקוד הכי רחוק - והווקטור מהמרכז אליו יהיה הציר הראשון.
- מחפשים את הקודקוד הכי רחוק מהציר הזה - והווקטור מהציר אליו יהיה הציר השני.
- הציר השלישי מחושב מהשניים האחרים באמצעות מכפלה ווקטורית.

השיטה הזאת יותר מדוייקת מ-AABB, אבל הרבה יותר יקרה מבחינה חישובית. כדי לבדוק התנגשות בין שתי OBB, מחשבים את ההטלה על כל אחד מהצירים (XYZ):

$$r_a = a_1 |A^1 \cdot L| + a_2 |A^2 \cdot L| + a_3 |A^3 \cdot L|$$

ואם $|T \cdot L| > r_a + r_b$ אז אין חפיפה. T זה הווקטור בין מרכזי הקופסאות, אבל מה זה L ? איזה צירים לבדוק? יש 15 צירים - 3 הצירים הראשיים של כל קופסא ועוד הצירים של האלכסונים. זה הרבה צירים, ואם לא חייבים עדיף לא להשתמש בשיטה הזאת.

Bounding Sphere Sweep Test

אובייקטים יכולים לזוז מהר בין פריימים, ו"לקפוץ" אחד דרך השני. אנחנו רוצים, למרות זה, לזהות שהם התנגשו. נגדיר:

$$A(u) = A_0 + uv_a \quad B(u) = B_0 + uv_b$$

כאשר:

$$v_a = A_1 - A_0 \quad v_b = B_1 - B_0 \quad 0 \leq u \leq 1$$

האובייקטים מתנגשים כאשר:

$$[B(u) - A(u)] \cdot [B(u) - A(u)] = (r_a + r_b)^2$$

מקבלים משוואה ריבועית עם עד שני פתרונות. אם אין לה פתרון (או שאף פתרון לא בתחום $0 \leq u \leq 1$) אז אין התנגשות. אם יש שני פתרונות בתחום את לוקחים את המינימלי.

תגובה להתנגשות

זיהינו התנגשות - מה עכשיו?

- האפשרות הכי פשוטה - לבטל את התנועה. להישאר במקום.

- חוקים פיסיקליים:

- חישוב התגובה לפי חוקי ניוטון.

- חישוב לפי חוק הוק: $F = -kx$ (זה קבוע הקפיץ, x זה displacement)

- חישוב כוח המשיכה

- חיכוך: $F_f = \mu F$

אופטימיזציות

כאשר מזיזים דברים צריך להתחיל לבדוק התנגשויות. אם בודקים בין כל אובייקט לכל אובייקט אחר צריך לבדוק $O(N^2)$ התנגשויות - וזה הרבה, אז רוצים לעשות אופטימיזציות.

אופטימיזציה פשוטה - אם יש לנו במודל שלנו חדרים, אז אובייקט בחדר אחד לא יכול להתנגש עם אובייקטים בחדרים אחרים. לכן אם מראש נבנה את העולם בצורה היררכית, יהיה לנו יותר קל.

אפשר גם להשתמש ב-BSP¹ - אם כל האובייקטים שלנו מסודרים בצורה כזאת בעץ, אז אפשר לבדוק לכל אובייקט איפה הוא בכל צד של כל מישור. אם אנחנו בצד אחד של המישור, אין מה לבדוק התנגשויות עם אובייקטים שנמצאים בצד השני.