**Collision – Supporting document**

Weaknesses of using Axis-Aligned Bounding Boxes (AABB)

Despite tests for AABB intersections being cheap, these bounding boxes won’t help when doing pairwise tests for every single possible pair in time. There are ways to minimize the number of tests done (such as space partitioning) and speed up the process but since the AABBs are to be moving most of the time, more and more indices must be created to keep up. In short, AABBs can end up causing lag and memory problems that – when used on a higher scale game – can end up with the program being slow, janky and maybe even make it crash.

Another problem that comes up when using AABB – especially in SFML – is that rotating sprites make the AABB larger and may cause unintentional effects to happen. To expand the paragraph above more; when using AABB in programs for fast rotating boxes in 3D, the best work-around calculation for this new rotation will either lag the system intensely (by repeating a calculation every frame) or only have to be done sparingly which would then decrease the bounding box’s accuracy.

Alternative collision solutions

To avoid these problems, I will be showing two other collision methods; Those being Bounding Circles and Oriented Bounding boxes (OBBs).

Bounding Circles – like all collision methods – have advantages and disadvantages but when applied to the right situations can improve on the AABB collisions. With Bounding Circles; There’s no need to rotate the shape and therefore do not suffer from calculations being done every frame through rotation. In addition, they are pretty easy to understand as calculations and quite efficient when being run by a program. However, some shapes don’t fit well into a circle (things like flat ground and platforms) and – even though the bounds don’t need to be rotated – the sprites still need to rotate, creating accuracy problems as some shape change a lot when rotated into a different angle.

On the other hand, Oriented Bounding Boxes also don’t need to recalculate their vertices every single frame and instead just change their corners. Unlike both Axis-Aligned Bounding Boxes and Bounding Circles, the calculations are costlier for the computer. The real advantage of OBBs come from the fact that you can calculate the AABB for less compactional power and then transform the AABBs into OBBs in order to make recalculating the box much less of a problem for the program to handle. The main downside of this is that it is less accurate than the AABB and OBB method as separate boxes.

Informed end decision

After looking into all of these different types of collisions, I have decided to use AABB in my players and tile platforms, as these objects will not rotate and do not fit into a circle. I will also be using Bounding Circles for the grenade and grenade launcher objects as these will be able to fit into circles and may and up rotating throughout gameplay. I have chosen not to use OBBs as this program will be small and any lag that OBBs would prevent will be barely noticeable to the player.