Overall, our group did a great job of overcoming different obstacles and pushing through to collaborate to get our project all put together.

First, we did a great job splitting up the work evenly and sticking to the planned divisions. We also did a great job of supporting each other through different challenges we faced. For example, if one person was struggling with a small part of their project feature, they would mention it in our discord and we would work together to research it or think through it so that we could move past those obstacles efficiently. Additionally, we did a great job of catching up with each other and implementing changes together. We had frequent calls to catch up on what they were working on, where each person was in their goals, and blockers that we could work through together. This allowed us to easily keep all our versions on the same page rather than trying to merge a bunch of random features together at the end.

On the other hand, we had a few struggles but not too many. The main struggle was trying to implement things before thinking through the details first. This caused us to have certain features take longer than expected (and probably needed) such as the basic movement and the camera. The basic movement was implemented but in a weird way that was causing bugs throughout the program until we eventually recognized the need to implement it in the suggested clean way that fixed a lot of bugs. The camera seemed straightforward until there were many small details that were being buggy and took quite a while to work through. Both of these are just examples of different areas that should've been researched very thoroughly and thought about before trying to implement because it just ended up being implemented with bugs and having to go back and fix it anyways.

Overall, our group did a great job of balancing the workload, working throughout the week instead of cramming, keeping each other updated, and collaborating in our work but had a bit of struggles with having things planned out before trying to implement them.

Code Examples:

```
// Sound initialization
void Cactus::initialize_sound() {
   String hurt_path = "res://audio/hurt.mp3";
   Ref<FileAccess> hurt_file = FileAccess::open(hurt_path, FileAccess::ModeFlags::READ);
   FileAccess *hurt_ptr = Object::cast_to<FileAccess>(*hurt_file);
   hurt = memnew(AudioStreamMP3);
   hurt->set_data(hurt_ptr->get_file_as_bytes(hurt_path));
   sound_effects = get_node<AudioStreamPlayer>("AudioStreamPlayer");
}

// Enabling value to be editable in the editor
void Camera::_bind_methods() {
   ClassDB::bind_method(D_METHOD("get_speed"), &Camera::get_speed);
   ClassDB::bind_method(D_METHOD("set_speed", "speed"), &Camera::set_speed);
   ClassDB::add_property("Camera", PropertyInfo(Variant::FLOAT, "camera sensitivity",
         PROPERTY_HINT_RANGE,  "0.0, 900.0, 1.0"), "set_speed", "get_speed");
}

// Use mouse motion to rotate camera around player in rotate mode and rotate camera and player when in
// strafe mode
const InputEventMouseMotion *key_event = Object::cast_to<const InputEventMouseMotion>(*event);
         if (key_event) {
                Node3D* tgt = Object::cast_to<Node3D>(get_parent());
                Player* player = get_node<Player>("../../../Player");
                if (!player->get_ad_rotate()) {
                        rotation[0] = 0.0;
                        rotation += Vector3(0.0, -1 * key_event->get_relative()[0] / (1000 - speed), 0.0);
                        player->set_rotation(rotation);
                } else {
                        rotation += Vector3(0.0, -1 * key_event->get_relative()[0] / (1000 - speed), 0.0);
                        tgt->set_rotation(rotation);
                }
        }
}


// dithering for camera collisions
if (camera_cast1->is_colliding() && camera_cast2->is_colliding()) {
        colliding = Object::cast_to<Node3D>(camera_cast1->get_collider());
        colliding->set_visible(false);
}
if (!camera_cast1->is_colliding() && !camera_cast2->is_colliding() && colliding) {
        colliding->set_visible(true);
        colliding = NULL;
}

// The player gathering food to increase their lives
// After the player collects the food, it is moved to a new random position on the map
bool entered = food1->is_entered() || food2->is_entered() ||
        food3->is_entered() || food4->is_entered();
        if (entered && Input::get_singleton()->is_action_just_pressed("E")) {
                if (!interact_player->is_playing() && !mute_sound_effects) {
                        play_interact();
                }
                lives++;
                if (food1->is_entered()) {
                        food1->set_position(Vector3(rand.randf_range(-50, 50), rand.randf_range(2, 20),
                                rand.randf_range(-50, 50)));
                emit_signal("interact_orange");
                }
```