



UNIVERSIDAD
DE GRANADA

ETSIIT, Facultad de Ciencias

DOBLE GRADO EN ING. INFORMÁTICA Y MATEMÁTICAS

TRABAJO DE FIN DE GRADO

Visualización de Superficies 3D

Presentado por:

Daniel Zufri Quesada

Tutor:

Carlos Ureña Almagro

Departamento de Lenguajes y Sistemas Informáticos

Pedro A. García Sánchez

Departamento de Álgebra

Curso académico 2022-2023

Índice general

1	Función Distancia con Signo	1
1.1	Operaciones sobre SDF	2
1.2	Raymarching y Spheretracing	7
1.3	Iluminación	8
1.3.1	Cálculo del vector normal	9
1.4	Aproximación de implícitas	10
2	Implicitación	13
2.1	Polinomios en varias variables	13
2.2	Bases de Groebner	14
2.3	Implicitación Polinomial	18
3	Desarrollo de la aplicación	19
3.1	Librería de polinomios multivariable	19
3.1.1	Clase Monomial	19
3.1.2	Clase Polynomial	19
3.1.3	Clase Ideal	19
	Bibliografía	21

1 Función Distancia con Signo

Tenemos como objetivo representar superficies en \mathbb{R}^3 . En general, estas pueden estar definidas de multitud de formas, siendo una de las más usuales a través de una función implícita. Nosotros nos centraremos en un tipo especial de funciones implícitas, que introducimos a continuación.

Definición 1.1. Sea $\Omega \subseteq \mathbb{R}^3$. Una **función distancia** es aquella que a cada punto de \mathbb{R}^3 le asigna su menor distancia a la frontera de Ω :

$$d_\Omega: \mathbb{R}^3 \rightarrow \mathbb{R}_0^+.$$
$$x \mapsto \inf\{\|x - y\| : y \in \partial\Omega\}.$$

Cuando Ω sea cerrado, podremos usar el mínimo en lugar del ínfimo.

Definición 1.2 (SDF). Sea $\Omega \subseteq \mathbb{R}^3$. Una **función distancia con signo** es una función de la forma:

$$\phi_\Omega: \mathbb{R}^3 \rightarrow \mathbb{R}.$$
$$x \mapsto \begin{cases} d_\Omega(x), & x \in \mathbb{R}^3 \setminus \overset{\circ}{\Omega}. \\ -d_\Omega(x), & x \in \overset{\circ}{\Omega}. \end{cases}$$

En general, nos referiremos a esta función por sus siglas en inglés SDF (*Signed Distance Function*), y la denotaremos simplemente ϕ siempre que no haya confusión.

Definición 1.3. Dada una función $\phi: \mathbb{R}^3 \rightarrow \mathbb{R}$ y $k \in \mathbb{R}$, llamamos **isosuperficie de ϕ con valor k** al conjunto:

$$S_{\phi,k} = \{(x, y, z) : \phi(x, y, z) = k\}.$$

Sin pérdida de generalidad podemos suponer $k = 0$, pues de no ser el caso, tomamos la función $\phi'(x, y, z) = \phi(x, y, z) - k$ y tenemos que $S_{\phi',0} = S_{\phi,k}$. Por tanto, la denotaremos como S_ϕ .

Nuestra intención será entonces construir una escena definida como la isosuperficie generada por un SDF. A partir de ahora, tomaremos $p = (x, y, z) \in \mathbb{R}^3$.

Ejemplo 1.1. Ejemplos simples de SDF ϕ en p para diferentes conjuntos Ω son:

- **Esfera de radio r centrada en el origen.**

$$\Omega = \{x \in \mathbb{R}^3 : \|x\| = r\}, \quad \phi(p) = \|p\| - r.$$

- **Plano con vector normal unitario $n = (a, b, c)$ y pasando por el origen.**

$$\Omega = \{p \in \mathbb{R}^3 : ax + by + cz = 0\}, \quad \phi(p) = p \cdot n.$$

- Toro de radios R y r , con $R > r$:

$$\Omega = \left\{ p \in \mathbb{R}^3 : \left(R - \sqrt{x^2 + y^2} \right)^2 + z^2 = r^2 \right\}, \quad \phi(p) = \left\| (\|(x, 0, z)\| - R, y) \right\| - r.$$

1.1. Operaciones sobre SDF

Si bien estas primitivas son fáciles de generar, también son muy simples y nos serán insuficientes si queremos construir escenas más complejas. Una de las ventajas de los SDF es que se pueden generar nuevas formas a partir de primitivas de forma muy sencilla. Para ello, una de las técnicas más útiles es la geometría de sólidos constructiva, que utiliza operaciones booleanas para combinar múltiples primitivas. Por la naturaleza de los SDF, estas operaciones se implementan fácilmente usando las funciones máx y mín.

Definición 1.4 (Operaciones Booleanas). Sean A y B isosuperficies generadas por ϕ y γ respectivamente. Definimos los SDF para las siguientes operaciones.

- **Unión:** $\text{sdf}_{A \cup B}(p) = \min(\phi(p), \gamma(p))$,
- **Intersección:** $\text{sdf}_{A \cap B}(p) = \max(\phi(p), \gamma(p))$,
- **Diferencia:** $\text{sdf}_{A \setminus B}(p) = \max(\phi(p), -\gamma(p))$.

Observación 1.1. Solo en el caso de la unión se obtiene un SDF según lo establecido en la Def. 1.2, ya que al aplicar máx en el interior (donde $\phi < 0$), las funciones resultado pueden ser solo una cota superior de la distancia. En nuestro caso solo estamos interesados en visualizar la frontera de las superficies, así que podemos obviar este problema.

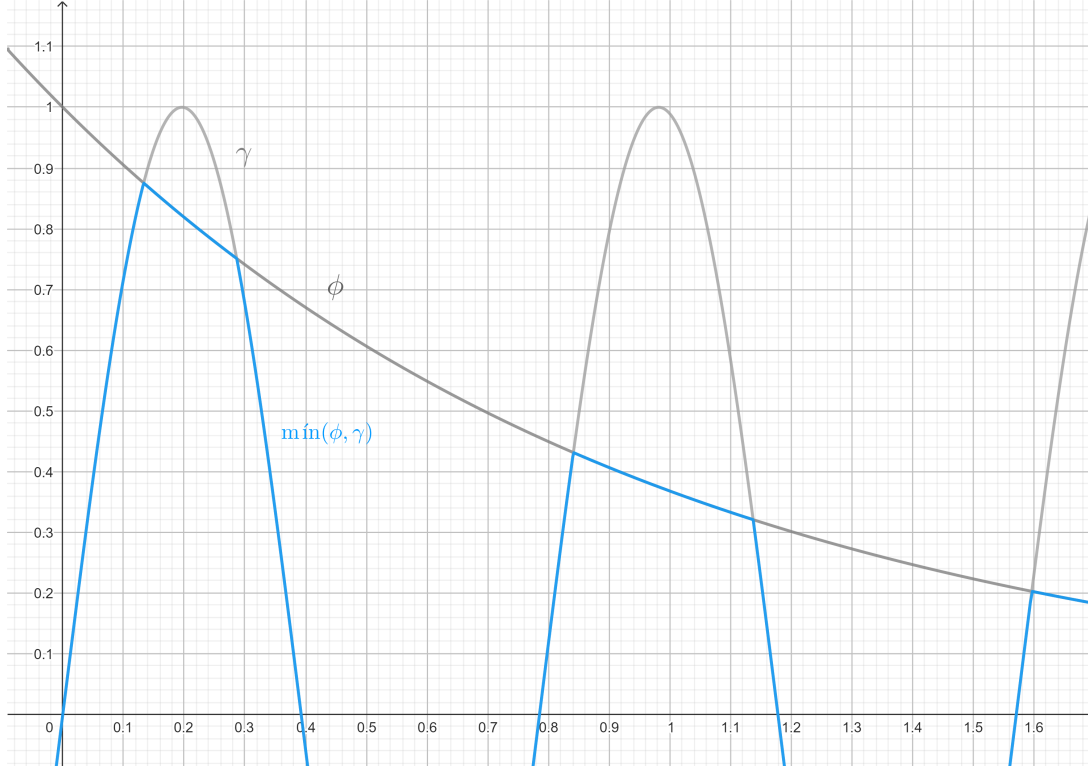
Un problema de usar estas transformaciones es que produce discontinuidades en la derivada del SDF resultante. Trataremos de evitar esta situación, además de por motivos analíticos, por motivos visuales, ya que el resultado de estas operaciones produce bordes muy acusados en la intersección de ambas superficies. Existen muchas formas de combinar SDF de forma más natural. Usaremos una de las más extendidas, usada por programas de modelado 3D como Blender [1] o videojuegos como [2], y ha sido estudiada por Íñigo Quilez en su web [3].

Observación 1.2. Para mayor claridad, en las figuras se representarán funciones de variable real, a pesar de que en nuestro caso sería en \mathbb{R}^3 .

Empezamos centrándonos en la unión. La idea es, dadas ϕ y γ , añadir una corrección para cada punto a la función mín original para que cumpla ciertos requisitos. Además esta función tendrá un parámetro $k \in \mathbb{R}$ que permita controlar su influencia. Llamaremos a esta corrección $\omega: \mathbb{R}^3 \times \mathbb{R}_0^+ \rightarrow \mathbb{R}$, de forma que la versión suavizada de mín será:

$$\begin{aligned} \text{smín}: \mathbb{R}^3 \times \mathbb{R}_0^+ &\rightarrow \mathbb{R}. \\ (p, k) &\mapsto \min(\phi(p), \gamma(p)) - \omega(p, k). \end{aligned}$$

Como no queremos que este cambio afecte al algoritmo de *raymarching*, debe cumplirse $\min(p) \geq \text{smín}(p) \forall p \in \mathbb{R}^3$, esto es, $\omega(p) \geq 0 \forall p \in \mathbb{R}^3$. Si estudiamos como se comporta la versión real de mín en la Figura 1.1, vemos que hay que corregirla cerca de las intersecciones de ϕ y γ , es decir, cuando ϕ y γ estén arbitrariamente cerca. En el resto de puntos no

Figura 1.1: Gráfica de $\min: \mathbb{R} \rightarrow \mathbb{R}$

queremos modificar la función original, luego estudiamos el comportamiento de smín en la bola cerrada centrada en p de radio k :

$$B_{p,k} = \{p \in \mathbb{R}^3 : |\phi(p) - \gamma(p)| \leq k\}, \quad p \in \mathbb{R}^3, \quad k \in \mathbb{R}_0^+$$

siendo $\omega(p) = 0$ cuando $p \notin B_{p,k}$.

Como queremos que smín sea continua en la frontera de $B_{p,k}$, imponemos la condición $\omega(p) = 0 \quad \forall p \in \delta B_{p,k}$. Por otro lado, es lógico que ω alcance su valor máximo en el punto de intersección, luego imponemos también $\omega(p) = s$, donde $s \in \mathbb{R}$ es el máximo de ω en $B_{p,k}$, y deberemos ajustarlo más adelante. Con estos requisitos, tenemos una primera aproximación para ω dado $p \in B_{p,k}$:

$$\omega(p,k) = s \left(1 - \frac{|\phi(p) - \gamma(p)|}{k} \right)^n = \begin{cases} s \left(1 - \frac{\phi(p) - \gamma(p)}{k} \right)^n, & \phi(p) > \gamma(p) \\ s \left(1 + \frac{\phi(p) - \gamma(p)}{k} \right)^n, & \phi(p) \leq \gamma(p) \end{cases}, \quad s \in \mathbb{R}, \quad n \in \mathbb{N},$$

donde hemos añadido el parámetro n para añadir más control al resultado final. Es evidente que está bien definida, pues:

$$\phi(p) = \gamma(p) \implies \frac{\phi(p) - \gamma(p)}{k} = 0 \implies \omega(p) = s$$

1 Función Distancia con Signo

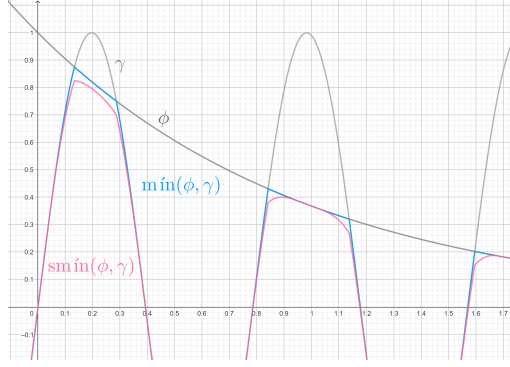


Figura 1.2: $k = 0.6$

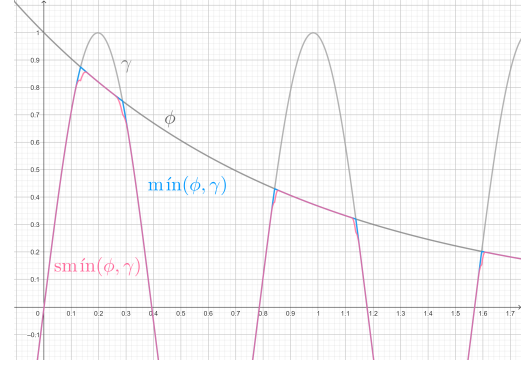


Figura 1.3: $k = 0.1$

Figura 1.4: Primera aproximación de $smín(p, k)$ con $s = 0.05$ y $n = 2$

Pasamos ahora a solucionar el problema inicial, el de la continuidad de la derivada.

$$\begin{aligned} smín'(p, k) &= (\min(\phi(p), \gamma(p)))' - \omega(p, k)' \\ &= \begin{cases} \gamma'(p) + sn \left(1 - \frac{\phi(p) - \gamma(p)}{k}\right)^{n-1} \left(\frac{\phi'(p) - \gamma'(p)}{k}\right), & \phi(p) > \gamma(p) \\ \phi'(p) - sn \left(1 + \frac{\phi(p) - \gamma(p)}{k}\right)^{n-1} \left(\frac{\phi'(p) - \gamma'(p)}{k}\right), & \phi(p) \leq \gamma(p) \end{cases} \end{aligned}$$

Dado $c \in \mathbb{R}^3$ tal que $\phi(p) = \gamma(p)$, veamos cuando está bien definida:

$$\phi' - sn \left(1 + \frac{\phi - \gamma}{k}\right)^{n-1} \left(\frac{\phi' - \gamma'}{k}\right) = \gamma' + sn \left(1 - \frac{\phi - \gamma}{k}\right)^{n-1} \left(\frac{\phi' - \gamma'}{k}\right)$$

Evaluando en c :

$$\begin{aligned} \cancel{\phi'(c)} - \cancel{\gamma'(c)} &= 2sn \left(1 + \frac{\phi(c) - \gamma(c)}{k}\right)^{n-1} \left(\frac{\cancel{\phi'(c)} - \cancel{\gamma'(c)}}{k}\right); \\ s &= \frac{k}{2n} \left(1 - \frac{\phi(c) - \gamma(c)}{k}\right)^0; \\ s &= \frac{k}{2n} \end{aligned}$$

Hemos llegado a la expresión final

$$\begin{aligned} \omega(p, k) &= \begin{cases} \frac{k}{2n} \left(1 - \frac{|\phi(p) - \gamma(p)|}{k}\right)^n, & |\phi(p) - \gamma(p)| \leq k, \\ 0, & \text{otro caso} \end{cases} \\ &= \frac{\max(k - |\phi(p) - \gamma(p)|, 0)^n}{2n \cdot k^{n-1}}, \quad s \in \mathbb{R}, \quad n \in \mathbb{N}, \end{aligned}$$

Podemos observar los resultados en la **Figura 1.7**.

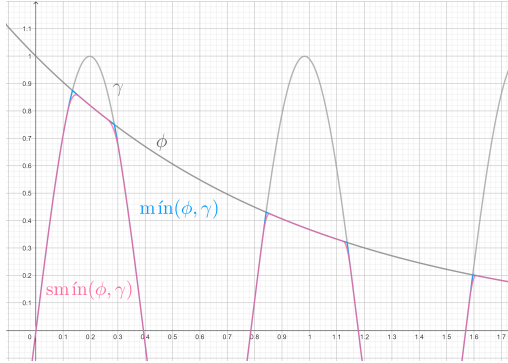


Figura 1.5: $k = 0.1, n = 2$

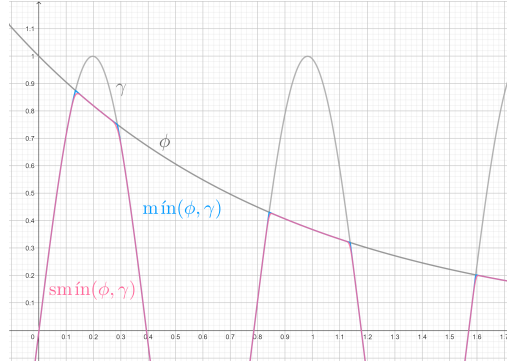


Figura 1.6: $k = 0.1, n = 3$

Figura 1.7: Resultado final de $smin(p, k)$

Finalmente, para obtener una versión suavizada del máximo, es fácil comprobar que

$$smáx(\phi, \gamma, k) = -smín(-\phi, -\gamma, k)$$

Recogemos los resultados obtenidos a continuación.

Definición 1.5 (Operaciones Booleanas Suavizadas). Sean A y B isosuperficies generadas por ϕ y γ respectivamente. Definimos los SDF para las operaciones suavizadas como sigue.

- **Unión suavizada:** $sdf_{unionS}(p) = \min(\phi(p), \gamma(p)) - \frac{\max(k - |\phi(p) - \gamma(p)|, 0)^n}{2n \cdot k^{n-1}}$,
- **Intersección suavizada:** $sdf_{interS}(p) = -\min(-\phi(p), -\gamma(p)) + \frac{\max(k - |\phi(p) - \gamma(p)|, 0)^n}{2n \cdot k^{n-1}}$,
- **Diferencia suavizada:** $sdf_{difS}(p) = -\min(-\phi(p), \gamma(p)) + \frac{\max(k - |\phi(p) + \gamma(p)|, 0)^n}{2n \cdot k^{n-1}}$,

donde $k \in \mathbb{R}_0^+$ es un coeficiente del suavizado.

Observamos que los operadores definidos en la **Def. 1.4** no son más que un caso particular de estos últimos cuando $k = 0$.

Pasamos ahora a estudiar otro tipo de operaciones, que nos resultarán útiles para generar nuevas formas a partir de un único SDF. Todas ellas se basarán en, dada ϕ , aplicar una transformación $t : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ a cada punto de la isosuperficie S_ϕ para obtener una nueva S_γ . Si queremos saber si un punto $q \in \mathbb{R}^3$ está en S_γ , tenemos que comprobar si su preimagen por la transformación pertenece a S_ϕ . Por tanto, bastará evaluar el SDF original en $t^{-1}(p)$, obteniendo que

$$\gamma(p) = \phi(t^{-1}(p))$$

Esto funciona bien para transformaciones como las traslaciones o rotaciones, ya que mantienen las distancias. Sin embargo, este no es el caso del escalado, ya que si tomamos $l(p) = sp$ con $s \in \mathbb{R}_0^+$:

$$\|p - p'\| = l(d) \implies \|l(p) - l(p')\| = \|sp - sp'\| = s\|p - p'\| = s \cdot d, \text{ donde } p, p' \in A$$

Como las distancias se escalan, deberemos hacer lo propio con el nuevo SDF, aplicándole el mismo factor de escalado s como muestra la [Def. 1.6](#).

Definición 1.6 (Operaciones afines). Sea A una isosuperficie. Definimos los SDF para las siguientes operaciones.

- **Traslación de vector v :** $\text{sdf}_{\text{traslacion}}(p) = \text{sdf}_A(p - v)$,
- **Escalado uniforme de dimensiones s :** $\text{sdf}_{\text{escalado}}(p) = \text{sdf}_A(p/s) \cdot s$,
- **Rotaciones de ángulo α sobre los ejes x, y, z :**

$$\text{sdf}_{\text{rotX}}(p) = \text{sdf}_A(R_x^{-1}(\alpha) \cdot p^t), \quad R_x(\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{pmatrix},$$

$$\text{sdf}_{\text{rotY}}(p) = \text{sdf}_A(R_y^{-1}(\alpha) \cdot p^t), \quad R_y(\alpha) = \begin{pmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{pmatrix},$$

$$\text{sdf}_{\text{rotZ}}(p) = \text{sdf}_A(R_z^{-1}(\alpha) \cdot p^t), \quad R_z(\alpha) = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Siguiendo el mismo razonamiento, podemos definir operaciones que modifiquen la geometría de la superficie.

Definición 1.7 (Operaciones Deformantes). Sea A una isosuperficie. Definimos los SDF para las siguientes operaciones.

- **Torsión:** $\text{sdf}_{\text{torsion}}(p) = \text{sdf}_A(p')$, con $p' = R_z(ky) \cdot (x, z, y)^t$,
- **Plegado:** $\text{sdf}_{\text{plegado}} = \text{sdf}_A(p')$, con $p' = R_z(kx) \cdot p^t$,
- **Redondeo:** $\text{sdf}_{\text{redondeo}}(p) = \text{sdf}_A(p) - k$,
- **Desplazamiento:** $\text{sdf}_{\text{desplazamiento}}(p) = \text{sdf}_A(\delta(p))$.

donde

- $k \in \mathbb{R}_0^+$ es un coeficiente de deformación,
- $\delta: \mathbb{R}^3 \rightarrow \mathbb{R}^3$ es un patrón de desplazamiento,
- $R_z(\alpha) \in \mathcal{M}_3(\mathbb{R})$ es la matriz de rotación de ángulo α sobre el eje z dada en la [Def. 1.6](#).

Introducimos por último operaciones que permiten generar copias de otras superficies.

Definición 1.8 (Operadores de Posicionamiento). Sea A una isosuperficie. Definimos los SDF para las siguientes operaciones.

- **Simetrías sobre los ejes x, y, z :**

$$\begin{aligned} \text{sdf}_{\text{simX}}(p) &= \text{sdf}_A(|x|, y, z), & \text{sdf}_{\text{simY}}(p) &= \text{sdf}_A(x, |y|, z), \\ \text{sdf}_{\text{simZ}}(p) &= \text{sdf}_A(x, y, |z|), \end{aligned}$$

- **Simetrías sobre los planos xy, xz, yz :**

$$\text{sdf}_{\text{simXY}}(p) = \text{sdf}_A(|x|, |y|, z), \quad \text{sdf}_{\text{simXZ}}(p) = \text{sdf}_A(|x|, y, |z|),$$

$$\text{sdf}_{\text{simYZ}}(p) = \text{sdf}_A(x, |y|, |z|),$$

- **Repetición $l \in \mathbb{N}^3$ veces en los ejes x, y, z con separación $s \in \mathbb{R}$:**

$$\text{sdf}_{\text{rep}}(p) = \text{sdf}_A(p - s \cdot c(r(p/s)), -l, l),$$

- **Repetición infinita:**

$$\text{sdf}_{\text{repInf}}(p) = \text{sdf}_A\left(\left(p + \frac{l}{2} \bmod l\right) - \frac{l}{2}\right)$$

donde:

- $c: \mathbb{R} \times \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$, $c(x, a, b) = \min(\max(x, a), b)$ acota x en $[a, b]$,
- $r: \mathbb{R}^3 \rightarrow \mathbb{R}^3$ redondea las componentes de un vector a sus enteros más cercanos.

Observación 1.3. Hay casos en los que los SDF definidos en la [Def. 1.8](#) podrían no ser exactos, al igual que ocurriría con la intersección y la diferencia en la [Def. 1.4](#):

1. Para las simetrías, cuando el objeto interseca el plano de simetría,
2. Para las repeticiones, cuando las dimensiones del objeto sean mayores o iguales a $l/2$.

A la vista de todas estas operaciones a nuestra disposición, es evidente el potencial que tienen los SDF, tanto en la generación y modificación de figuras como en su eficiencia, ya que podemos visualizar miles de objetos al precio de uno utilizando las simetrías o repeticiones.

1.2. Raymarching y Spheretracing

Una vez definida la escena necesitamos un método para visualizarla. Utilizaremos la técnica de *raymarching*. Para ello deberemos elegir un punto donde estará el observador. A partir de él se trazarán rayos hacia cada uno de los puntos de un plano imaginario que llamaremos plano de visión, de forma que cada punto de este plano corresponde a un pixel de la imagen resultante. Si el rayo interseca con S_ϕ significa que ese pixel corresponde a un punto de S_ϕ y será coloreado como tal.

Para comprobar esta intersección utilizamos un método iterativo: a partir de la posición del observador, en cada iteración avanzamos en la dirección del rayo una distancia fija δ . Evaluamos entonces nuestro SDF en la posición actual. Si este valor es muy cercano a 0 significará que estamos en la isosuperficie. De lo contrario, repetiremos el proceso hasta encontrar la intersección o superar el número máximo de iteraciones, en cuyo caso concluiremos que no hay intersección.

TODO: diagrama

Sin embargo podemos mejorar esta técnica, ya que cuanto más alejados estén los puntos de S_ϕ del observador, mayor es el número de iteraciones necesarias para encontrar la intersección. Esto es debido a que si no queremos perder ninguna intersección, el valor de incremento δ deberá de ser pequeño. La solución a este problema es el uso de *spheretracing*. Su funcionamiento es similar al *raymarching*, con la diferencia de que el incremento en la posición del rayo no es fija, sino que es la máxima que podemos tomar asegurándonos de no perder información en cada iteración. Esta distancia será entonces la distancia mínima del punto actual del rayo a S_ϕ , que no es más que evaluar ϕ en dicho punto.

TODO: diagrama de pérdida de intersección
El algoritmo sería por tanto el descrito en 1.

Algorithm 1: Spheretracing

Data: origen p , dirección v
 incremento $\leftarrow 0$;
 distanciaTotal $\leftarrow 0$;
for $i \in \text{MAX_ITERACIONES}$ **do**
 rayo $\leftarrow p + \text{distanciaTotal} \cdot v$
 distanciaActual $\leftarrow \phi(\text{rayo})$
 if distanciaActual $< \epsilon$ **then**
 return DibujarSuperficie(rayo);
 end
 distanciaTotal $+= \text{distanciaActual}$
 if distanciaTotal $> \text{MAX_DISTANCIA}$ **then**
 return DibujarFondo();
 end
end

1.3. Iluminación

Ya sabemos qué píxeles pertenecen a la superficie, pero no de qué color deben dibujarse. Para ello utilizaremos el modelo de reflexión de Phong. En él trabajaremos con los siguientes parámetros:

- Luz ambiente i_a : vector RGB suma de las contribuciones de todas las fuentes de luz de la escena. Contiene un valor por defecto que será el color del fondo.
- Lista de fuentes de luz $(l_1, \dots, l_m, \dots, l_n)$.
- Materiales asignados a cada objeto definidos por las constantes:
 - Reflexión especular k_s : modela cómo se refleja la luz en objetos brillantes. Depende de tanto de la posición de las fuentes de luz como de la del observador.
 - Reflexión difusa k_d : modela cómo se refleja la luz en objetos mates. Depende de la posición de las fuentes de luz, pero no de la posición del observador.
 - Reflexión ambiental k_a : representa la proporción de luz del ambiente que refleja el objeto.

- Coeficiente de brillo α : valor real positivo que permite variar el tamaño de las zonas brillantes (a mayor valor, menor tamaño y más pulida o especular).
- Los vectores normalizados definidos para cada punto de la superficie p :
 - L_m : vector director desde p a cada fuente de luz l_m .
 - N vector normal a la superficie en p .
 - R_m dirección del rayo de luz reflejado especularmente desde la fuente l_m en el punto p .
 - V : dirección de p a la posición del observador.

TODO: diagrama vectores

Con estas variables, el color en p vendrá dado por la fórmula:

$$I_p = k_a i_a + \sum_{m=0}^n (k_d (L_m \cdot N) + k_s (R_m \cdot V)^\alpha)$$

De los vectores anteriores, L_m y V se calculan trivialmente, y R_m se puede obtener como

$$R_m = 2(L_m \cdot N)N - L_m$$

Veamos como obtener N .

1.3.1. Cálculo del vector normal

Definición 1.9. El **gradiente** de una función implícita $\phi: \mathbb{R}^3 \rightarrow \mathbb{R}$ es $\nabla\phi = \left(\frac{\partial\phi}{\partial x}, \frac{\partial\phi}{\partial y}, \frac{\partial\phi}{\partial z} \right)$.

Proposición 1.1. $\nabla\phi$ es perpendicular a S_ϕ .

Demostración. Sea $s \in S_\phi$ arbitrario. Tomamos una curva parametrizada:

$$\begin{aligned} \alpha: [0, 1] &\rightarrow S_\phi \\ t &\mapsto (x(t), y(t), z(t)) \end{aligned}$$

cumpliendo $\alpha(t_0) = s$. Veamos que $\nabla\phi(s) \perp \alpha$:

$$\begin{aligned} \alpha(t) \subset S_\phi &\implies \phi(\alpha(t)) = 0 \\ \implies \nabla\phi(\alpha(t)) &= \frac{\partial F}{\partial x} \frac{dx}{dt} + \frac{\partial F}{\partial y} \frac{dy}{dt} + \frac{\partial F}{\partial z} \frac{dz}{dt} = 0 \\ \implies \langle \nabla\phi(x, y, z), \alpha'(t) \rangle &= 0 \implies \langle \nabla\phi(x, y, z), \alpha'(t_0) \rangle = 0 \end{aligned}$$

Por tanto, $\nabla\phi(s)$ es perpendicular al vector tangente de α en s , que a su vez está contenida en el plano tangente de S_ϕ en s . Por tanto $\nabla\phi(s) \perp S_\phi$. \square

Hemos visto que calcular N equivale a calcular $\nabla\phi$. Sin embargo, dado que la superficie viene dada por un SDF que podría ser no derivable, no podemos hacerlo de forma analítica. De ser derivable, podría ser una buena opción calcular el gradiente una única vez al momento de definir la ϕ , tras lo cual para obtener N solo habría que realizar evaluaciones de dicho gradiente. Sin embargo, por su sencillez y popularidad, nos decantaremos por un método numérico que únicamente hará uso de evaluaciones de ϕ .

Definición 1.10. Dada $\phi : \mathbb{R}^3 \rightarrow \mathbb{R}$ diferenciable, $p = (x, y, z)$, $v \in \mathbb{R}^3$, definimos la **derivada direccional** en p con dirección v a:

$$\nabla_v f(p) = \nabla f(p) \cdot v = \frac{\partial f(p)}{\partial x} v_x + \frac{\partial f(p)}{\partial y} v_y + \frac{\partial f(p)}{\partial z} v_z$$

Para el cálculo de cada parcial de f podemos utilizar la definición de derivada. Por ejemplo, para la primera componente:

$$\frac{\partial f(p)}{\partial x} v_x = \lim_{h \rightarrow 0} \frac{f(p + (h, 0, 0)) - f(p)}{h} v_x$$

Procedemos a calcular N aplicaremos el **método del tetraedro**, el cual se basa en evaluar ϕ en la dirección de los vértices de un tetraedro:

$$k_0 = (1, -1, -1) \quad k_1 = (-1, -1, 1) \quad k_2 = (-1, 1, -1) \quad k_3 = (1, 1, 1)$$

Proposición 1.2 (Método del tetraedro). Dado $p \in S_\phi$, su vector normal N se obtiene normalizando el vector

$$\hat{N} = \sum_{i=0}^3 k_i \cdot f(p + hk_i)$$

Demostración. Por la proposición **Proposición 1.1**, basta comprobar que \hat{N} es colineal a $\nabla \phi(p)$.

$$\begin{aligned} \hat{N} &= \sum_{i=0}^3 k_i \cdot f(p + hk_i) = \sum_{i=0}^3 k_i \cdot f(p + hk_i) + k_i \cdot f(p) = \sum_{i=0}^3 k_i \cdot f(p + hk_i - f(p)) = \sum_{i=0}^3 k_i \nabla_{k_i} f(x) \\ &= \sum_{i=0}^3 k_i \cdot (k_i \cdot \nabla f(p)) = \sum_{i=0}^3 (k_i \cdot k_i) \nabla f(p) = \sum_{i=0}^3 \nabla f(p) = 4 \nabla f(p) \end{aligned}$$

donde hemos usado que $\sum_{i=0}^3 k_i = (0, 0, 0)$, $\sum_{i=0}^3 k_i \cdot k_i = (1, 1, 1)$, y que el producto escalar es un operador lineal. \square

1.4. Aproximación de implícitas

Empezábamos el capítulo diciendo que una de las representaciones más comunes de superficies es a través de implícitas, pero hasta ahora nos hemos centrado en estudiar un tipo particular de este tipo de ecuaciones. Si intentásemos aplicar los algoritmos anteriores a una función implícita cualquiera, observaríamos que el resultado tiene ciertos defectos, tales como deformaciones, o incluso no se visualiza. Vamos a introducir una técnica que dada una función implícita ϕ , nos permitirá obtener un SDF aproximado de S_ϕ . Esto nos será útil cuando no conozcamos o no podamos calcular explícitamente el SDF de una superficie pero sí su representación implícita.

Proposición 1.3. Sea $\phi : \mathbb{R}^3 \rightarrow \mathbb{R}$ una función implícita cualquiera y $p \in \mathbb{R}^3$. Entonces:

$$sdf_{S_\phi}(p) \approx \frac{|\phi(p)|}{|\nabla \phi(p)|}$$

Demostración. Sea q al punto de S_ϕ más cercano a p y $v = \vec{pq} \perp S_\phi$ en q . Queremos calcular la distancia de p a S_ϕ , que es justamente v . Podemos calcular el desarrollo de Taylor de ϕ en p :

$$\phi(p+v) = \phi(p) + \nabla(p) \cdot (p+v-p) + \mathcal{O}(|p+v-p|^2) \implies \phi(p+v) = \phi(p) + \nabla(p) \cdot v + \mathcal{O}(|v|^2)$$

Suponemos ahora que p está cerca de S_ϕ , esto es, $|v| < \varepsilon$. Como $\phi(p+v) = \phi(q) = 0$, tenemos que:

$$\begin{aligned} 0 = |\phi(p+v)| &\approx |\phi(p) + \nabla(p) \cdot v| \leq |\phi(p)| + |\nabla(p) \cdot v| \\ &\leq |\phi(p)| + |\nabla(p)| |v| \implies |v| \leq \frac{|\phi(p)|}{|\nabla(p)|} \end{aligned}$$

donde hemos usado la desigualdad triangular y las propiedades del producto escalar. \square

Al igual que nos ocurría al calcular el vector normal, habrá ocasiones en las que el gradiente no pueda ser obtenido de forma analítica. Esta vez sin embargo no nos basta con conocer solo la dirección del gradiente, así que no podemos aplicar de nuevo el método del tetraedro. Usaremos un nuevo método en su lugar que usa la definición de derivada.

Proposición 1.4 (Método de las diferencias centrales). Sea $\phi: \mathbb{R}^3 \rightarrow \mathbb{R}$. Entonces:

$$\frac{\partial \phi}{\partial x}(p) \approx \frac{\phi(p + (h, 0, 0)) - \phi(p - (h, 0, 0))}{2h}$$

$$\frac{\partial \phi}{\partial y}(p) \approx \frac{\phi(p + (0, h, 0)) - \phi(p - (0, h, 0))}{2h}$$

$$\frac{\partial \phi}{\partial z}(p) \approx \frac{\phi(p + (0, 0, h)) - \phi(p - (0, 0, h))}{2h}$$

donde $h \in \mathbb{R}^+$ es un valor arbitrariamente pequeño.

Para calcular el gradiente necesitaremos evaluar ϕ un total de seis veces, ya que hay que realizar dos evaluaciones por cada variable. Es por este motivo que

2 Implicitación

Ahora que sabemos representar las superficies generadas por una función implícita cualquiera, nos proponemos ser capaces de representar también superficies definidas paramétricamente. Para ello haremos uso de la teoría de bases de Groebner, pero antes debemos introducir una serie de definiciones y resultados previos que nos pongan en contexto.

2.1. Polinomios en varias variables

Fijamos A cuerpo y $X = \{x_1, \dots, x_n\}$ un conjunto de variables distintas.

Definición 2.1. Llamamos **monomio** en X al producto de la forma:

$$x_1^{\alpha_1} \cdots x_n^{\alpha_n}, \quad \alpha_i \in \mathbb{N}, \quad \forall 0 \leq i \leq n$$

Lo denotaremos como X^α , y diremos que $\alpha \in \mathbb{N}^n$ es el **exponente** del monomio.

Definición 2.2. Definimos un **polinomio** en X con coeficientes en A a toda combinación lineal finita de monomios:

$$\sum_{\alpha \in \mathbb{N}^n} a_\alpha X^\alpha, \quad a_\alpha \in A$$

Proposición 2.1. El conjunto de polinomios, que denotaremos $A[X] = A[x_1, \dots, x_n]$, es un cuerpo con las operaciones:

- *Suma heredada de A :* $(f + g)(\alpha) = f(\alpha) + g(\alpha), \quad \forall f, g \in A[X].$
- *Producto de convolución:* $(fg)(\alpha) = \sum_{\beta + \gamma = \alpha} f(\beta)g(\gamma), \quad \forall f, g \in A[X].$

Nos surge la pregunta de si hay alguna forma “natural” de ordenar los monomios que forman un polinomio, al igual que ocurre para polinomios de una sola variable. Primero tenemos que especificar qué es lo que consideramos “natural”.

Definición 2.3. Un **orden admisible** es un orden total \leq sobre \mathbb{N}^n cumpliendo:

1. $(0, \dots, 0) \leq \alpha, \quad \forall \alpha \in \mathbb{N}^n.$
2. $\alpha < \beta \implies \alpha + \gamma < \beta + \gamma, \quad \forall \alpha, \beta, \gamma \in \mathbb{N}^n.$

Siempre supondremos que usamos un orden admisible, luego podemos ordenar los monomios que conforman un polinomio ordenando sus exponentes según cualquier dicho orden. Hay multitud de órdenes entre los que elegir. Algunos de los más importantes son:

Ejemplo 2.1. Definimos el **orden lexicográfico** \leq_{lex} como:

$$\alpha \leq_{\text{lex}} \beta \iff \begin{cases} \alpha = \beta \\ \text{ó} \\ \alpha_i < \beta_i, \text{ donde } i \text{ es el primer índice tal que } \alpha_i \neq \beta_i \end{cases}$$

Definición 2.4. Llamamos **grado** de $\alpha \in \mathbb{N}^n$ a $|\alpha| = \sum_{i=1}^n \alpha_i + \dots + \alpha_n$.

Ejemplo 2.2. Definimos el **orden lexicográfico graduado inverso** $\leq_{\text{degrevlex}}$ como:

$$\alpha \leq_{\text{degrevlex}} \beta \iff \begin{cases} |\alpha| < |\beta| \\ \text{ó} \\ |\alpha| = |\beta| \text{ y } \alpha_i > \beta_i, \text{ donde } i \text{ es el último índice tal que } \alpha_i \neq \beta_i \end{cases}$$

Una vez obtenida la noción de orden admisible, estamos en disposición de definir varios conceptos que nos resultarán imprescindibles.

Definición 2.5. Sea $f = \sum_{\alpha \in \mathbb{N}^n} a_{\alpha} X^{\alpha}$ y \leq admisible. Definimos:

- **Exponente:** $\exp(f) = \max_{\leq} \alpha$.
- **Monomio líder:** $\text{lm}(f) = X^{\exp(f)}$
- **Coefficiente líder:** $\text{lc}(f) = a_{\exp(f)}$.
- **Término líder:** $\text{lt}(f) = \text{lc}(f) \cdot \text{lm}(f)$.

Teorema 2.1. Sea $F = \{f_1, \dots, f_s\} \subset A[X]$. Entonces todo polinomio $f \in A[X]$ se puede expresar como:

$$f = q_1 f_1 + \dots + q_s f_s + r$$

donde $q_i, r \in A[X]$ y r no se puede expresar como combinación lineal de ningún elemento de F o es 0. Notaremos $\text{rem}(f, [F]) = r$.

En otras palabras, podemos dividir f entre los polinomios f_1, \dots, f_s para expresar f como combinación lineal de elementos de F . Sin embargo, esta descomposición no será única, y dependerá del orden que ocupen los polinomios en F . Podemos calcular esta descomposición usando **Algoritmo 2**.

Demostración. TODO

□

2.2. Bases de Groebner

Definición 2.6. Dado un anillo conmutativo R , decimos que $\emptyset \neq I \subseteq R$ es un **ideal** si:

1. $a + b \in I, \forall a, b \in I$.
2. $ab \in I, \forall a \in I, \forall b \in R$.

Lo denotaremos como $I \leq R$.

Definición 2.7. Dado $F \subseteq A$, el **ideal generado** por F es:

$$\langle F \rangle = \{a_1 f_1 + \dots + a_s f_s : a_1, \dots, a_s \in A, f_1, \dots, f_s \in F\}$$

Definición 2.8. Dado $I \leq A[X]$, diremos que $G = \{g_1, \dots, g_s\} \subseteq I$ es una **base de Groebner** para I si $\langle \text{lt}(I) \rangle = \langle \text{lt}(g_1), \dots, \text{lt}(g_s) \rangle$.

Algorithm 2: División polinomios varias variables

```

Data:  $f, F = [f_1, \dots, f_s]$ 
 $p \leftarrow f$ ;
 $[q_1, \dots, q_s] \leftarrow [0, \dots, 0]$ ;
 $r \leftarrow 0$ ;
while  $p \neq 0$  do
    divisorEncontrado  $\leftarrow false$ ;
    for  $f_i \in F$  do
        if  $\exp(p) = \exp(f_i) + \alpha$  then
             $q_i \leftarrow q_i + \frac{\text{lc}(p)}{\text{lc}(f_i)} X^\alpha$ ;
             $p \leftarrow p - f_i \cdot \frac{\text{lc}(p)}{\text{lc}(f_i)} X^\alpha$ ;
            divisorEncontrado  $\leftarrow true$ ;
        end
    end
    if !divisorEncontrado then
         $r \leftarrow r + \text{lt}(p)$ ;
         $p \leftarrow p - \text{lt}(p)$ ;
    end
end
return  $[r, q_1, \dots, q_s]$ 

```

Definición 2.9. Dada $f \in A[X]$, definimos su **soporte** como

$$\text{supp}(f) = \{\alpha \in \mathbb{N}^n : f(\alpha) \neq 0\}$$

Definición 2.10. G se dirá una **base de Groebner reducida** para I si para todo $g \in G$ se cumple:

- $\text{lc}(g) = 1$.
- $\text{supp}(g) \cap (\exp(G \setminus \{g\}) + \mathbb{N}^n) = \emptyset$.

La base de Groebner es a un ideal lo que un sistema de generadores a un espacio vectorial: un subconjunto a partir del cual podemos obtener el total. Si además la base es reducida, esta será el equivalente a la base del espacio vectorial. Por tanto, resulta de gran interés saber si dado un ideal I siempre existirá una base de Groebner asociada, y de ser así, si podemos calcularla.

Definición 2.11. Dados $\alpha, \beta \in \mathbb{N}^n$, definimos:

- **mínimo común múltiplo:** $\text{lcm}(\alpha, \beta) = \{\text{máx}(\alpha_1, \beta_1), \dots, \text{máx}(\alpha_n, \beta_n)\}$.
- **máximo común divisor:** $\text{gcd}(\alpha, \beta) = \{\text{mín}(\alpha_1, \beta_1), \dots, \text{máx}(\alpha_n, \beta_n)\}$.

Definición 2.12. Sean $f, g \in A[X]$ donde $\alpha = \exp(f)$, $\beta = \exp(g)$, $\gamma = \text{lcm}(\alpha, \beta)$. Definimos el **S-polinomio** de f y g como

$$S(f, g) = \text{lc}(g)X^{\gamma-\alpha}f - \text{lc}(f)X^{\gamma-\beta}g$$

Teorema 2.2 (Primer Criterio de Buchberger). Sean $I \leq A[X]$ y $G = \{g_1, \dots, g_t\}$ conjunto de generadores de I . Entonces:

$$G \text{ es base de Groebner para } I \iff \text{rem}(S(g_i, g_j), G) = 0, \forall 1 \leq i < j \leq t$$

El algoritmo que usaremos para el cálculo de la base de Groebner se basará en este criterio. Sin embargo, antes de presentarlo, estudiamos dos criterios adicionales que lo harán más eficiente evitando el cálculo de S-polinomios innecesarios, que es la operación más costosa del algoritmo al suponer la realización de numerosas divisiones.

Teorema 2.3 (Criterios de Buchberger). Sean $I \leq A[X]$ y $g_1, g_2 \in G \subseteq A[X]$. Si se cumple cualquiera de las siguientes condiciones:

1. $\text{lcm}(g_1, g_2) = \text{lm}(g_1)\text{lm}(g_2)$.
 2. $\exists f \in G: \text{lm}(f) | \text{lcm}(g_1, g_2)$ y se verifica alguna de las condiciones:
 - a) algún $S(g_i, f) \xrightarrow{G} 0$.
 - b) $\text{lm}(f) | \frac{\text{lm}(g_i)}{\text{gcd}(g_1, g_2)}$ y $\text{sm}(g_i)\text{lm}(f) \neq \text{sm}(f)\text{lm}(g_i)$.
- donde $i, j \in \{1, 2\}$ e $i \neq j$.

Entonces $S(g_1, g_2) \xrightarrow{G} 0$

Aplicando estos criterios, obtenemos **Algoritmo 3**.

Algorithm 3: Algoritmo de Buchberger optimizado

Data: $f, F = [f_1, \dots, f_s]$
 $G \leftarrow F$;
repeat
 $G' \leftarrow G$;
 for each pair $\{f, g\} \subseteq G'$ **do**
 if !Criterio 1(f, g) **AND** !Criterio 2(f, g, G') **then**
 $r \leftarrow \text{rem}(S(f, g), G')$;
 if $r \neq 0$ **then**
 $G \leftarrow G \cup \{r\}$;
 end
 end
 end
until $G' = G$;
return G

Observación 2.1. Una decisión necesaria y que aún no hemos mencionado es la de cómo se eligen las parejas $\{f, g\}$. Uno de los métodos más usados es la conocida como **estrategia normal**, debido a su simpleza y haber probado ser de las que completan más rápido el algoritmo. Esta consiste en tomar el par f, g cuyo $\text{lcm}(f, g)$ sea del menor grado posible según el orden admisible usado.

Ahora ya somos capaces de calcular una base de Groebner de todo ideal $I \leq A[X]$. No obstante, nos gustaría que siempre que fuera posible, esta fuera reducida. Veamos que siempre existe esta base reducida y cómo obtenerla.

Definición 2.13. Sea $M \leq \mathbb{N}^n$. Decimos que A es un **conjunto generador minimal** de M si:

$$M = a + \mathbb{N}^n \quad \wedge \quad M \neq (A \setminus \{\alpha\}) + \mathbb{N}^n, \quad \forall \alpha \in A$$

Lema 2.1. *Todo ideal tiene un único conjunto generador minimal.*

Teorema 2.4. *Todo ideal I admite una única base de Groebner reducida para un orden admisible dado.*

Demostración. Existencia Sea G un conjunto generador minimal de $\exp(I)$, que sabemos que existe por **Lema 2.1**. Sea $g \in G$ y $r = \text{rem}(g, [G \setminus \{g\}])$. Tenemos que:

$$\exp(g) \notin \exp(G \setminus \{g\}) + \mathbb{N}^n \implies \exp(g) = \exp(r) \implies \exp(G) = \exp((G \setminus \{g\}) \cup \{r\})$$

Como $g - r \in \langle G \setminus \{g\} \rangle \subseteq I$, tenemos que $r \in I$. Por tanto $G' = (G \setminus \{g\}) \cup \{r\}$ es base de Groebner de I y cumple $\text{supp}(r) \cap (\exp(G' \setminus \{r\}) + \mathbb{N}^n) = \emptyset$. Aplicando este procedimiento a cada $g \in G$ obtenemos una base reducida de I .

Unicidad Sean G_1, G_2 dos bases minimales de I . Por **Lema 2.1**:

$$\exp(G_1) = \exp(G_2) \implies \forall g_1 \in G_1, \exists! g_2 \in G_2 : \exp(g_1) = \exp(g_2)$$

Como $g_2 - g_1 \in I \implies \text{rem}(g_1 - g_2, G_1) = 0$. Por otro lado:

$$\left. \begin{aligned} \text{supp}(g_1 - g_2) &\subseteq (\text{supp}(g_1) \cup \text{supp}(g_2)) \setminus \{\exp(g_1)\} \\ \text{supp}(g_i) \setminus \{\exp(g_i)\} \cap (\{\exp(G_i) + \mathbb{N}^n\}) &= \emptyset, \quad i \in \{1, 2\} \end{aligned} \right\} \implies \text{supp}(g_1 - g_2) \cap (\exp(G_1) + \mathbb{N}^n) = \emptyset$$

$$\implies \text{rem}(g_1 - g_2, G_1) = g_1 - g_2 \implies g_1 = g_2 \implies G_1 = G_2$$

□

De esta demostración podemos obtener **Algoritmo 4** para reducir una base de Groebner cualquiera a su respectiva minimal:

Algorithm 4: Minimización de base de Groebner

Data: G base a minimizar

$G \leftarrow F;$

foreach $g \in G$ **do**

$g \leftarrow g/\text{lc}(g);$

$r \leftarrow \text{rem}(g, [G \setminus \{g\}]);$

if $r \neq 0$ **then**

$g \leftarrow r;$

end

end

2.3. Implícitación Polinomial

Definición 2.14. Dado $F \subseteq A[X]$, llamamos **variedad afín** definida por F al conjunto:

$$\mathbb{V}(F) = \{(a_1, \dots, a_n) \in A^n : f_i(a_1, \dots, a_n) = 0, \forall 1 \leq i \leq s\}$$

Definición 2.15. Dado $I \leq A[x_1, \dots, x_n]$, diremos que su **ideal de l -eliminación** es:

$$I_l = I \cap A[x_{l+1}, \dots, x_n] \leq A[x_{l+1}, \dots, x_n]$$

Definición 2.16. Decimos que un orden admisible \leq es un **orden de l -eliminación** si

$$\beta \leq \alpha \implies \beta \in \mathbb{N}_l^n, \forall \alpha \in \mathbb{N}_l^n, \forall \beta \in \mathbb{N}^n$$

donde $\mathbb{N}_l^n = \{\alpha \in \mathbb{N}^n : \alpha_i = 0, 1 \leq i \leq l\}$.

Teorema 2.5 (Eliminación). Sea $I \leq A[x_1, \dots, x_n]$ y G una base de Groebner suya respecto a un orden \leq de l -eliminación. Entonces, una base de Groebner para I_l viene dada por:

$$G_l = G \cap A[x_{l+1}, \dots, x_n]$$

Teorema 2.6 (Implícitación Polinomial). Dados $f_1, \dots, f_n \in A[t_1, \dots, t_r]$ con A cuerpo infinito, sea

$$\begin{aligned} \phi: A^r &\rightarrow A^n \\ (a_1, \dots, a_r) &\mapsto (f_1(a_1, \dots, a_r), \dots, f_n(a_1, \dots, a_r)) \end{aligned}$$

Definimos los ideales:

- $I = \langle x_1 - f_1, \dots, x_n - f_n \rangle \leq A[t_1, \dots, t_r, x_1, \dots, x_n]$
- $J = I \cap A[x_1, \dots, x_n]$ el ideal de r -eliminación de I

Entonces, $\mathbb{V}(J)$ es la menor variedad que contiene a $\phi(A^r)$.

3 Desarrollo de la aplicación

Nos centramos ahora en los aspectos de la implementación de una aplicación que haga uso de todas las técnicas y conceptos tratados en los capítulos anteriores. Para ello, se ha usado [React](#), una biblioteca de JavaScript (y TypeScript) para interfaces de usuario. Las principales ventajas de este enfoque son:

- La aplicación puede ser ejecutada en cualquier navegador, haciendo que sea mucho más accesible.
- React está basada en componentes modulares, lo que la hace escalable. Además, debido a su popularidad, hay una infinidad de librerías de terceros a nuestra disposición, ya sea específicas de React o de JavaScript.

Un aspecto fundamental a lo largo de todo el desarrollo será el del rendimiento. Esto es debido a que, por defecto, las aplicaciones web solo tienen a su disposición una hebra de ejecución (la de interfaz de usuario), haciendo de cuello de botella para el resto de cálculos.

3.1. Librería de polinomios multivariable

Si bien tenemos a nuestra disposición un gran número de librerías externas, en el momento de realización de la aplicación no encontré ninguna alternativa viable para trabajar con polinomios multivariable en JavaScript de forma nativa. Como alternativas se barajó el uso de la API de [Geogebra](#) o realizar llamadas a código Python que usara [SageMath](#). Sin embargo, por motivos de rendimiento y completitud, se decidió desarrollar una librería nativa en TypeScript para el manejo de polinomios en varias variables y cálculo de bases de Groebner. Se encuentra disponible en [GitHub](#) junto a su documentación, ejemplos de uso y tests usados.

La librería consta de tres clases que pasamos a estudiar a continuación.

3.1.1. Clase `Monomial`

3.1.2. Clase `Polynomial`

3.1.3. Clase `Ideal`

Bibliografía

- [1] B. Foundation, "Blender - 3d creation software - git repository," Último acceso: 6 mayo 2023.
- [2] M. Molecule, "Dreams," 2020.
- [3] I. Quílez, "Smooth minimun," [En línea]. Último acceso: 6 mayo 2023.