Rendszerarchitektúra: Qbit modell és UI integráció

1 Áttekintés

A rendszer két fő rétegből áll:

- **Adatréteg (Qbit modellek)**
- **UI réteg (DOM elemek + interakciók)**

A rétegeket folyamatos, kétirányú frissítés köti össze, amelyet főként a `refresh()` és különböző listenerek koordinálnak.

2 Adatréteg (Model) — Qbit

A **Qbit** a rendszer központi adatmodellje.

2.1 Főbb metódusok

- `refresh()` – gondoskodik a modell, a UI és a 3D scene szinkronizációjáról:
  1. Törli a régi 3D objektumot a scene-ből.
  2. Újrarajzolja a frissített vektort.
  3. Meghívja az UI frissítőt: `updateCoordinates(id, current)`.
- `reset()`, `remove()` és a kapu alkalmazása után mindig `refresh()` fut.

3 UI réteg — DOM elemek

A `createElement(...)` minden Qbithez létrehoz:

- Színdobozt
- Koordináta sort
- Reset és törlés gombot

Minden DOM elem az `id` alapján kapcsolódik a megfelelő Qbithoz a modellben. A DOM nem számol, csak kijelöl, indít, és tükrözi a modell változásait.

4 Listenerek

A listenerek a rendszer egyik legfontosabb részei, mivel összekapcsolják a felhasználói lépéseket a modell műveleteivel.

4.1 Regisztráció

A `RegisterListeners()` a betöltéskor hívódik meg, és az alábbi eseményeket köti:

- Kapu gombok (`X, Y, Z, H, S, Phase`)
- `Add` gomb
- `Clear all` gomb
- Téma váltó
- Koordináta inputok
- Reset/Törlés gombok és wrapper kattintások a DOM listaelemein

4.2 Model → UI visszacsatolás

Minden frissítés végén a listenerek **nem frissítik manuálisan a DOM-ot**. A modell maga kezeli a DOM frissítését, lentebb látható egy példa hogy hogyan:

- `refresh()` → `updateCoordinates(...)`

Így a UI soha nem írja felül hibásan a modell értékeit.

5 CSS rövid összefoglaló

A rendszer felhasználói felületét modern, két témás (light/dark) stílus jellemzi amiket futás közben tudunk változtatni:

- **Alap:** teljes képernyős layout, Arial betű, light/dark témák háttér- és színbeállításokkal.
- **UI konténer (#ui-container):** bal felső sarok, flex oszlop, lekerekített sarkok, árnyék, animált nyitás/bezárás.
- **Gombok:** .fancy-btn gradienttel, hover/active animáció, kapu gombok (X, Y, Z, H, S, PH input).
- **Listaelemek (#listContainer .element):** flex, lekerekített sarkok, témától függő színek, hover effekt, görgethető.
- **Műveletek:** .delete-btn és .reset-btn hover animáció, ikonok pointer-events: none.
- **Scene:** A Scene és a gömb a kiválasztott téma színeiben jelenik meg, és az egész képernyőt lefedi.

6 Matematikai modell és implementáció

6.1 Komplex számok

A kvantumállapotok és kapuk komplex amplitúdókkal dolgoznak. A megvalósításban minden komplex szám egy:

$$a = \{\text{re} : \alpha,\ \text{im} : \beta\}$$

objektum.
A fő műveletek:

- összeadás: $a + b$
- kivonás: $a - b$
- szorzás:
$$(a + ib)(c + id) = (ac - bd) + i(ad + bc)$$
- konjugálás: $a^* = \text{re} - i\,\text{im}$
- abszolút érték négyzete: $|a|^2 = \text{re}^2 + \text{im}^2$

Ezek építőkockaként szolgálnak minden vektor- és mátrixművelethez.

6.2 Kvantumállapot reprezentáció

Egy egyqubites állapot a szokásos bázisban:

$$|\psi\rangle = \begin{pmatrix}\alpha\\ \beta\end{pmatrix}, \qquad |\alpha|^2 + |\beta|^2 = 1.$$

A programban ez a következő formában jelenik meg:
`[C(re_alpha,im_alpha), C(re_beta,im_beta)]`

6.3 Bloch-gömb konverzió

A Bloch-gömb egy valós 3D vektort rendel minden tiszta egyqubites állapothoz:

$$x = 2\Re(\alpha^*\beta), \qquad y = 2\Im(\alpha^*\beta), \qquad z = |\alpha|^2 - |\beta|^2.$$

A következőkben lépésről lépésre elmagyarázzuk, hogyan és miért működik ez a leképezés, valamint hogyan lehet stabilan implementálni a gyakorlatban.

Miért létezik ez a leképezés?

Egy egyqubites állapotot felírhatunk

$$|\psi\rangle = \begin{pmatrix}\alpha\\ \beta\end{pmatrix}, \qquad |\alpha|^2 + |\beta|^2 = 1.$$

A kvantummechanika szerint a globális fázis nem mérhető, azaz az állapotok közti különbség csak az amplitúdók relatív fázisának és arányának függvénye. Ezt a két szabadságfokot a Bloch-gömb pontja (x, y, z) kifejezi: a gömb két szöge (θ, ϕ) egyértelműen leírja az állapotot a globális fázis figyelmen kívül hagyásával.

Állapot → Bloch:

Számítsuk ki a Bloch-koordinátákat:

$$\alpha^*\beta = |\alpha||\beta|e^{i(\arg\beta - \arg\alpha)}.$$

Ennek a valós és képzetes részei adják $x/2$ és $y/2$. A z komponens a bázisállapotok foglaltságkülönbségét méri:

$$z = |\alpha|^2 - |\beta|^2.$$

Ezek az összefüggések közvetlenül vezetnek a `stateToBloch()` implementációhoz: először kiszámoljuk $\alpha^*\beta$-t (komplex konjugál és szorzat), majd x, y, z-t a fenti formulák szerint, és visszaadjuk THREE.Vector3(x, y, z)-ként.

Bloch → állapot:

A Bloch-vektor pontjai a gömbön a szögekkel kapcsolatos egyszerű trigonometriai összefüggésekkel írhatók fel:

$$z = \cos\theta, \qquad x = \sin\theta\cos\phi, \qquad y = \sin\theta\sin\phi,$$

ahonnan

$$\theta = \arccos(z), \qquad \phi = \text{atan2}(y, x).$$

Ezek után egy kényelmes és gyakran használt választás (a globális fázis rögzítése érdekében) a következő állapot:

$$\alpha = \cos\frac{\theta}{2}, \qquad \beta = e^{i\phi}\sin\frac{\theta}{2}.$$

Ez kielégíti a normálási feltételt, és a relatív fázist ϕ-ként helyezi el a β komponensben.

Miért elég ez a formula?

A trigonometriai azonosságok miatt:

$$|\alpha|^2 = \cos^2\frac{\theta}{2} = \frac{1+\cos\theta}{2} = \frac{1+z}{2},$$

$$|\beta|^2 = \sin^2\frac{\theta}{2} = \frac{1-\cos\theta}{2} = \frac{1-z}{2},$$

és

$$2\Re(\alpha^*\beta) = 2\cos\frac{\theta}{2}\sin\frac{\theta}{2}\cos\phi = \sin\theta\cos\phi = x,$$

$$2\Im(\alpha^*\beta) = \sin\theta\sin\phi = y.$$

Tehát a fenti α, β visszavezeti pontosan az eredeti x, y, z-et (globális fázis nélkül).

6.4 Mátrixszorzás és normalizálás

A kvantumkapuk 2×2-es komplex mátrixok, amelyeket egy állapotra a szokásos mátrixszorzással alkalmazunk:

$$|\psi'\rangle = U|\psi\rangle.$$

A `matrixVectorMultiply()` függvény ezt általánosan kezeli. Minden eredmény végül normalizált lesz:

$$|\psi\rangle \leftarrow \frac{|\psi\rangle}{\sqrt{|\alpha|^2 + |\beta|^2}}.$$

6.5 Kvantumkapuk

- Pauli X:
$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$
- Pauli Y:
$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$
- Pauli Z:
$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$
- Hadamard:
$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$
- S-kapu (fázis $\pi/2$):
$$S = P\left(\tfrac{\pi}{2}\right) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/2} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$$
- T-kapu (fázis $\pi/4$):
$$T = P\left(\tfrac{\pi}{4}\right) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$$
- Általános fáziskapu:
$$P(\theta) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}$$

Minden kapu alkalmazásának lépései:

1. Bloch-vektor visszaalakítása kvantumállapottá.
2. Mátrixszorzás a kapumátrixszal.
3. Normalizálás.
4. Visszakonvertálás Bloch-vektorra.

6.6 Kapuk alkalmazása Bloch-gömbre

A 3D vizualizáció csak Bloch-koordinátákkal dolgozik, ezért a kapuk **Bloch → állapot → kapu → állapot → Bloch** módon működnek.

Például az X-kapu:

$$(x, y, z) \xrightarrow{\text{Bloch}\to\text{state}} |\psi\rangle \xrightarrow{X} X|\psi\rangle \xrightarrow{\text{state}\to\text{Bloch}} (x', y', z').$$

Ez biztosítja, hogy a 3D vizualizáció és a matematikai modell mindig pontos összhangban legyen.