DOI 10.34185/1562-9945-6-137-2021-14
УДК 004.9

А.А. Демчишин, Ю.С. Бурєнков

ВЕБ-СИСТЕМА МОДЕЛЮВАННЯ ПОВЕРХОНЬ НА ОСНОВІ ПАТЧЕЙ КЕТМАЛА-РОМА

Анотація. Однією з головних вимог, яка висувається до програмного забезпечення моделювання 3D об'єктів, є можливість вільно змінювати форму поверхні водночас з її проходженням через всі контрольні точки. В роботі отримано алгоритмічну модель побудови поверхні Ерміта з умовою Кетмала-Рома та ненульовими векторами кручення поверхні. Показано, що умова Кетмала-Рома дає можливість склеювати окремі патчі з гладкістю першого порядку, що є запорукою ергономічності поверхонь. Показано, що окремо взята контрольна точка має локальний вплив на поверхню, а саме на 12 навколишніх патчей. Розробка програмної системи моделювання поверхонь об'єктів з клієнтською частиною у вигляді веб-застосунку, який побудовано на основі архітектурного стилю Single-Page Application показала, що досвід користувача такого застосунку близький до досвіду користування десктопною програмою. В той же самий час SPA застосунок не потребує інсталяції та може з успіхом виконуватись, як на стаціонарних, так і на мобільних пристроях.
Ключові слова: поверхня Ерміта, сплайн Кетмала-Рома, NURBS, Single Page Application, гладкість поверхні.

Постановка проблеми. Сьогодні поверхні відіграють важливу роль в роботі конструкторів, вчених, художників, хірургів і інших людей, що пов'язані зі створенням інноваційних виробів. Розробка меблів, корпусів машин, телефонів, предметів одягу, будівель, навіть органів людини відбувається із залученням геометричного моделювання поверхонь [1,2]. На теперішній час існує багато методів побудови поверхонь, які дають змогу варіювати форму моделі на основі зміни позиції контрольних точок. До таких відносяться, наприклад, поверхні, що побудовано на основі параметричних кривих: тригонометричної кривої, кривої Безьє, B-сплайну, NURB-сплайну.

NURBS моделювання [3] – технологія неоднорідних раціональних B-сплайнів, створення плавних форм і моделей, у яких немає гострих країв, як у полігональних моделей. Саме через цю характеристику аналітичну модель

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

NURBS застосовують в системах моделювання Autodesk 3ds Max, Blender, Autodesk Maya, ZBrush та ін.

Короткий аналіз базових характеристик трійки найпопулярніших систем моделювання виглядає наступним чином. **Autodesk 3ds Max** – професійна програмна десктопна система 3D-моделювання, анімації, рендеринга візуальних ефектів і графіки [4]. Працює в операційних системах Microsoft Windows (32-біта/64-біта). Autodesk 3ds Max доступна в 2-ох ліцензійних версіях: студентська - безкоштовна версія програми яку, однак, не можна використовувати з метою отримання прибутку, а також повна - (комерційна) версія з вартістю підписки 1k € на рік.

Autodesk Maya – десктопний графічний редактор, для моделювання тривимірних об'єктів, анімації, композитингу та візуалізації. В даний час Autodesk Maya є найбільш розповсюдженою системою для розробки 3D графіки для кіно і телебачення. Спочатку розроблена для ОС IRIX (платформа SGI), була портована під ОС Linux, Microsoft Windows і Mac OS. З 2013 року версії програми випускаються тільки для 64-бітових систем. Вартість підписки 1k € на рік.

Blender – вільне і відкрите професійне десктопне програмне забезпечення для створення тривимірної комп'ютерної графіки, що включає в себе засоби моделювання, скульптингу, анімації, рендеринга, постобробки й монтажу відео зі звуком, компонування сцени за допомогою «вузлів» (node compositing), а також створення 2D-анімації. В даний час користується великою популярністю серед безкоштовних 3D-редакторів в зв'язку з його швидким і стабільним розвитком. Доступний для комп’ютера під керуванням Windows, macOS та Linux.

Узагальнюючою характеристикою наведених програмних систем є використання монолітного архітектурного стилю розробки програмного забезпечення, який є властивим для десктопних застосунків. Десктопні програмні системи вимагають інсталяції на локальний комп’ютер, що в свою чергу прив’язує користувача до певної операційної системи. Перевагою реалізації програмної систем у вигляді десктопного застосунку (рис. 1) є швидкість реалізації бізнес логіки впродовж перших років циклу розробки програмного продукту. До недоліків монолітного архітектурного стилю відноситься зростаюча складність системи за рахунок великої кількості модулів та нечітких залежностей між ними.

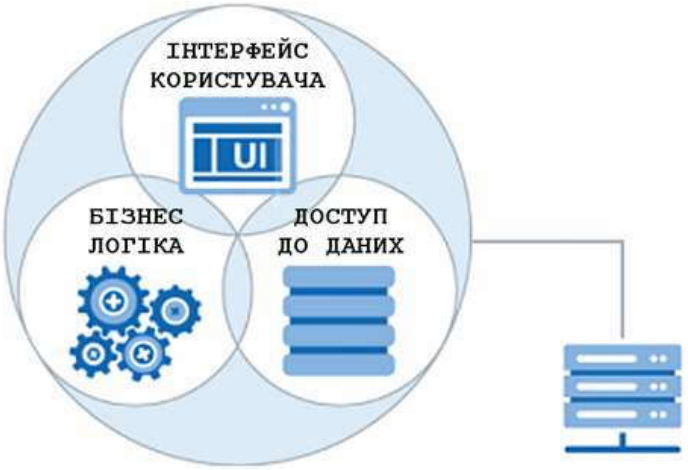

Рисунок 1 – Схема архітектури монолітного додатку

Аналіз останніх досліджень і публікацій. Останнім часом все більшу популярність серед розробників завойовує архітектурний стиль односторінкового інтерфейсу (Single-Page Application) – веб застосунку, який розташовано на одній сторінці [5]. SPA надає користувачеві досвід, близький до досвіду користування десктопною програмою, з іншого боку застосунок не потребує інсталяції.

З точки зору програмної реалізації, NURBS-поверхні, які використовуються в переліченому програмному забезпеченні, бувають двох видів: CV (Control Vertex)-поверхні та P (Point)-поверхні [6].

Особливістю NURBS CV-поверхні (1) є те, що вона не проходить обов'язково через всі контрольні точки, а натомість лише контролюється їх позицією (рис.2). При переміщенні контрольної точки, деяка частина поверхні, що лежить ближче до неї, тягнеться в напрямку зміщення, немов прикріплена пружиною. Така поведінка значно ускладнює конструювання. В той же час, хмара контрольних точок блокує частину екрану.

$$\mathbf{S}(u,v)=\frac{\sum_{i=0}^{m}\sum_{j=0}^{n}N_{i,p}(u)N_{j,q}(v)w_{i,j}\mathbf{P}_{i,j}}{\sum_{i=0}^{m}\sum_{j=0}^{n}N_{i,p}(u)N_{j,q}(v)w_{i,j}}, \qquad (1)$$

де $N_{i,p}(u), N_{j,q}(v)$ – B-сплайн базисні функції, $\mathbf{P}_{i,j}$ – радіус вектори контрольних точок, $w_{i,j}$ – ваги відповідних контрольних точок $\mathbf{P}_{i,j}$.

На відміну від CV-поверхонь, P-поверхні управляються вершинами, що знаходяться безпосередньо на самій поверхні (рис. 3). Така поведінка досягається за рахунок перерахунку переміщення окремої P точки у зміщення відповідної CV точки, прихованої від користувача. В свою чергу, оскільки зміна по-

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

зиції CV вершини впливає на всю поверхню, то ближні до неї P точки теж починають пливти, що ускладнює процес конструювання.

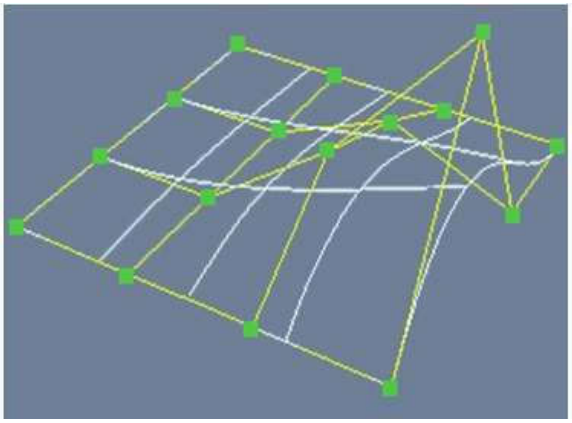

Рисунок 2 – NURBS поверхня типу CV

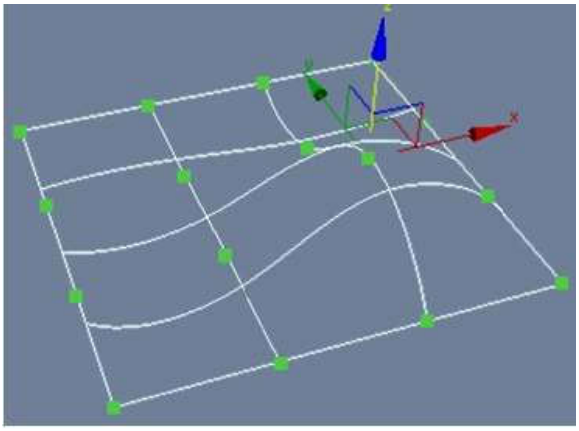

Рисунок 3 – NURBS поверхня типу Point

Виходячи з того, що NURBS є надмножиною багатьох різних методів сплайнів [7], використання NURBS в системах моделювання дає можливість експортувати моделі в інші системи.

Ваги $w_{i,j}$ дозволяють робити точний опис конічних поверхонь, і надають користувачеві більше контролю над формою об'єкту. Але такий контроль, на практиці, виявляється не дуже ефективним - хоча кожна точка має свою власну вагу, і кожна вага має локальний вплив на поверхню, зміна ваги однієї вершини призводить до зміни всієї поверхні. Стикування патчів NURBS поверхонь можливе за умови виконання низки обмежень [8,9].

Розвиток інформаційних процесів супроводжується появою нових вимог до алгоритмів моделювання. Однією з головних вимог, яка висувається до програмного забезпечення моделювання 3D об'єктів, є можливість вільно змінювати форму поверхні водночас з її проходженням через всі контрольні точки. Така вимога набула актуальності через необхідність часто вносити правки в форму виробу на етапі прототипування з урахуванням концепції "що бачиш, то і отримаєш" (what you see is what you get).

Мета дослідження. Виходячи з вищенаведеного актуальним є розробка алгоритмічної моделі конструювання такої поверхні, що: проходить через всі контрольні точки, кожна з точок має строго локальний вплив, окремі патчі автоматично стикуються з гладкістю C1. Для розширення аудиторії користувачів алгоритмічної моделі доцільним виглядає реалізація системи моделювання у вигляді веб-застосунку.

Викладення основного матеріалу дослідження. Розглянемо інтерполяцію положень двох точок $P(0), P(1)$ та двох напрямних векторів $P_t(0), P_t(1)$, що їм асоційовано (рис.4), вздовж параметричної кривої третього порядку у вигляді (5):

$$P(t) = B_1 + B_2 t + B_3 t^2 + B_4 t^3 . \quad (5)$$

Задамо граничні умови:

$$P(0) = B_1, P(1) = B_1 + B_2 + B_3 + B_4, \quad P_t(0) = B_2, P_t(1) = B_2 + 2B_3 + 3B_4,$$

та складемо систему лінійних рівнянь, з якої знайдемо невідомі коефіцієнти $B_{1..4}$. Підставимо коефіцієнти $B_{1..4}$ в (5), та запишемо рівняння сплайна Ерміта:

$$P(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P(0) \\ P(1) \\ P_t(0) \\ P_t(1) \end{bmatrix} \quad (7)$$

Функція (5) має неперервні першу та другу похідної між граничними точками (за визначенням).

Значення векторів похідної в точці $\overline{P}^i, i = 0,1,2\ldots$ будемо розраховувати як різницю координат наступної та попередньої контрольної точки сплайну (рис.5):

$$\overline{P}_t^i = (\overline{P}^{i+1} - \overline{P}^{i-1})/2 . \quad (4)$$

Умова (4) гарантує неперервність першої похідної в точці стиковки кусків сплайну.

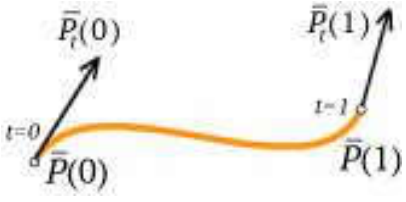

Рисунок 4 – Інтерполяція Ерміта

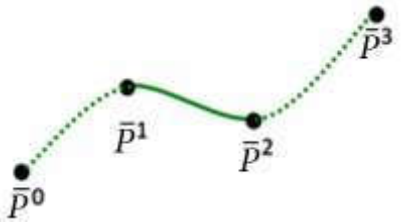

Рисунок 5 – Сплайн Кетмала Рома

В аналогічний спосіб отримаємо рівняння патча поверхні Ерміта:

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

$$P(u,v)=\begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix}\begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}\cdot$$
$$\cdot\begin{bmatrix} P(0,0) & P(0,0) & P_v(0,0) & P_v(0,1) \\ P(1,0) & P(0,0) & P_v(1,0) & P_v(1,1) \\ P_u(0,0) & P_u(0,1) & P_{uv}(0,0) & P_{uv}(0,1) \\ P_u(1,0) & P_u(1,1) & P_{uv}(1,0) & P_{uv}(1,1) \end{bmatrix}\begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}\begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix} \quad (15)$$

Вектори кручення $P_{uv}(u,v)$ розрахуємо за допомогою рівняння кінцевих різниць [10]:

$$P_{uv}^{i,j}=\frac{1}{4}(P^{i+1,j+1}-P^{i+1,j-1}-P^{i-1,j+1}+P^{i-1,j-1})$$

Алгоритмічна модель, що перетворює кортеж координат контрольних точок у тривимірні координати поверхні Ерміта з умовою Кетмала-Рома, представлено на рис. 6.

Реалізацію програмної системи (рис. 7) було проведено з використанням скриптової мови програмування JavaScript, бібліотеки 3D графіки Three.js, бази даних MongoDb та веб-фреймворку Express. Використання JavaScript продиктовано динамічним розвитком бібліотек з відкритим кодом, яке спостерігається після появи рушія Node.js, та швидкою еволюцією API браузеру Chrome. Залучення бібліотеки Three.js надало можливість проводити апаратно-пришвидшену візуалізацію 3D-моделей безпосередньо у веб-браузері, спираючись на стандарт WebGL (оснований на OpenGL ES 2.0).

Відхід від традиційної реляційної структури бази даних на користь NoSQL бази даних MongoDB, що побудовано навколо JSON-подібних документів з динамічними схемами, зробило можливим зберігання даних в форматі клієнтського застосунку без конвертації.

Графічний інтерфейс програмної системи представлено на рис.8,9. Інструменти інтерфейсу дозволяють додавати патчі до краю поверхні, переміщувати точки в площі екрану та вздовж осей координат.

```
F01: function MixedDerivative(i,j)
F02: {
F03:     let uv1 = mathjs.subtract(_2DArray[i+1][j+1], _2DArray[i+1][j-1]);
F04:     let uv2 = mathjs.subtract(_2DArray[i-1][j-1], _2DArray[i-1][j+1]);
F05:     return  mathjs.multiply(mathjs.add(uv1, uv2), 0.25);
F06: }

C01: function computePointOnSurface(uVal, vVal, i, j) {
C02:     let p1u = mathjs.subtract(_2DArray[i + 1][j],     _2DArray[i - 1][j]);
C03:     let p2u = mathjs.subtract(_2DArray[i + 1][j + 1], _2DArray[i - 1][j + 1]);
C04:     let p3u = mathjs.subtract(_2DArray[i + 2][j],     _2DArray[i][j]);
C05:     let p4u = mathjs.subtract(_2DArray[i + 2][j + 1], _2DArray[i][j + 1]);
C06:     p1u = mathjs.multiply(p1u, 0.5);
C07:     p2u = mathjs.multiply(p2u, 0.5);
C08:     p3u = mathjs.multiply(p3u, 0.5);
C09:     p4u = mathjs.multiply(p4u, 0.5);
C10:     let p1v = mathjs.subtract(_2DArray[i][j + 1],     _2DArray[i][j - 1]);
C11:     let p2v = mathjs.subtract(_2DArray[i][j + 2],     _2DArray[i][j]);
C12:     let p3v = mathjs.subtract(_2DArray[i + 1][j + 1], _2DArray[i + 1][j - 1]);
C13:     let p4v = mathjs.subtract(_2DArray[i + 1][j + 2], _2DArray[i + 1][j]);
C14:     p1v = mathjs.multiply(p1v, 0.5);
C15:     p2v = mathjs.multiply(p2v, 0.5);
C16:     p3v = mathjs.multiply(p3v, 0.5);
C17:     p4v = mathjs.multiply(p4v, 0.5);
C18:     if (!_2DArray[i-1][j-1] || !_2DArray[i-1][j+2] ||
C19:         !_2DArray[i+2][j-1] || !_2DArray[i+2][j+2]) {
C20:         var p1uv = new Array(0,0,0);
C21:         var p2uv = new Array(0,0,0);
C22:         var p3uv = new Array(0,0,0);
C23:         var p4uv = new Array(0,0,0);
C24:     }
C25:     else {
C26:         var p1uv = MixedDerivative(i,j);
C27:         var p2uv = MixedDerivative(i,j+1);
C28:         var p3uv = MixedDerivative(i+1,j);
C29:         var p4uv = MixedDerivative(i+1,j+1);
C30:     }
C31:     let mFu = new Array( 2.0 * Math.pow(uVal, 3) - 3.0 * Math.pow(uVal, 2) + 1.0,
C32:                         -2.0 * Math.pow(uVal, 3) + 3.0 * Math.pow(uVal, 2),
C33:                         Math.pow(uVal, 3) - 2.0 * Math.pow(uVal, 2) + uVal,
C34:                         Math.pow(uVal, 3) - Math.pow(uVal, 2));
C35:
C36:     let mFv = new Array( 2.0 * Math.pow(vVal, 3) - 3.0 * Math.pow(vVal, 2) + 1.0,
C37:                         -2.0 * Math.pow(vVal, 3) + 3.0 * Math.pow(vVal, 2),
C38:                         Math.pow(vVal, 3) - 2.0 * Math.pow(vVal, 2) + vVal,
C39:                         Math.pow(vVal, 3) - Math.pow(vVal, 2));
C40:     let p1 = _2DArray[i][j];
C41:     let p2 = _2DArray[i][j+1];
C42:     let p3 = _2DArray[i+1][j];
C43:     let p4 = _2DArray[i+1][j+1];
C44:     let mBx = mathjs.matrix([
C45:                             [p1[0], p2[0], p1v[0], p2v[0]],
C46:                             [p3[0], p4[0], p3v[0], p4v[0]],
C47:                             [p1u[0], p2u[0], p1uv[0], p2uv[0]],
C48:                             [p3u[0], p4u[0], p3uv[0], p4uv[0]]  ]);
C49:     let mBy = mathjs.matrix([
C50:                             [p1[1], p2[1], p1v[1], p2v[1]],
C51:                             [p3[1], p4[1], p3v[1], p4v[1]],
C52:                             [p1u[1], p2u[1], p1uv[1], p2uv[1]],
C53:                             [p3u[1], p4u[1], p3uv[1], p4uv[1]]  ]);
C54:     let mBz = mathjs.matrix([
C55:                             [p1[2], p2[2], p1v[2], p2v[2]],
C56:                             [p3[2], p4[2], p3v[2], p4v[2]],
C57:                             [p1u[2], p2u[2], p1uv[2], p2uv[2]],
C58:                             [p3u[2], p4u[2], p3uv[2], p4uv[2]]  ]);
C59:     let resX = mathjs.multiply(mathjs.multiply(mBx, mFv), mFu);
C60:     let resY = mathjs.multiply(mathjs.multiply(mBy, mFv), mFu);
C61:     let resZ = mathjs.multiply(mathjs.multiply(mBz, mFv), mFu);
C62:     return { xVal: resX, yVal: resY, zVal: resZ };
C63: }
```

Рисунок 6 – Алгоритмічна модель поверхні Ерміта з умовою Кетмала-Рома

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

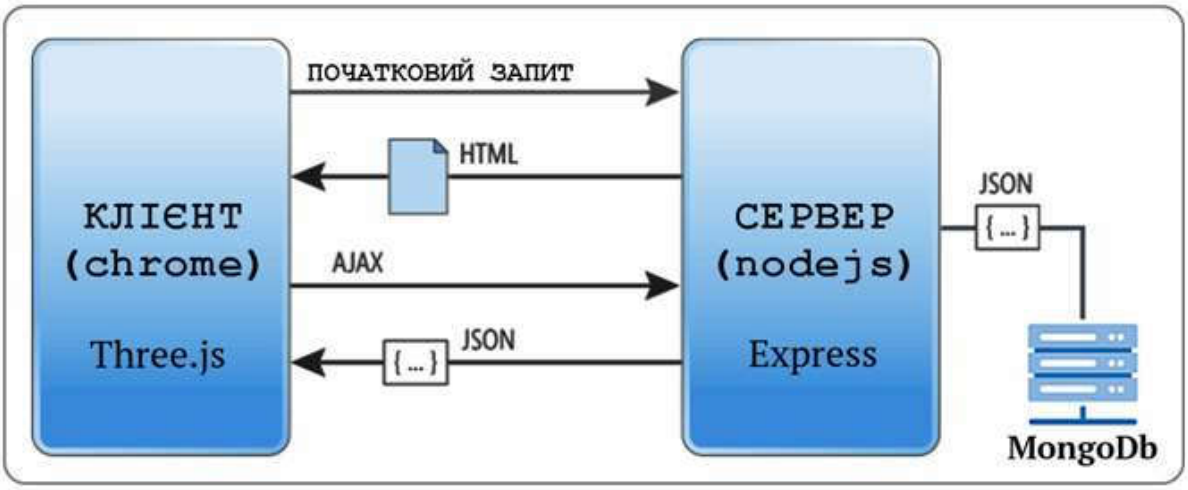

Рисунок 7 – Життєвий цикл веб-застосунку моделювання поверхонь

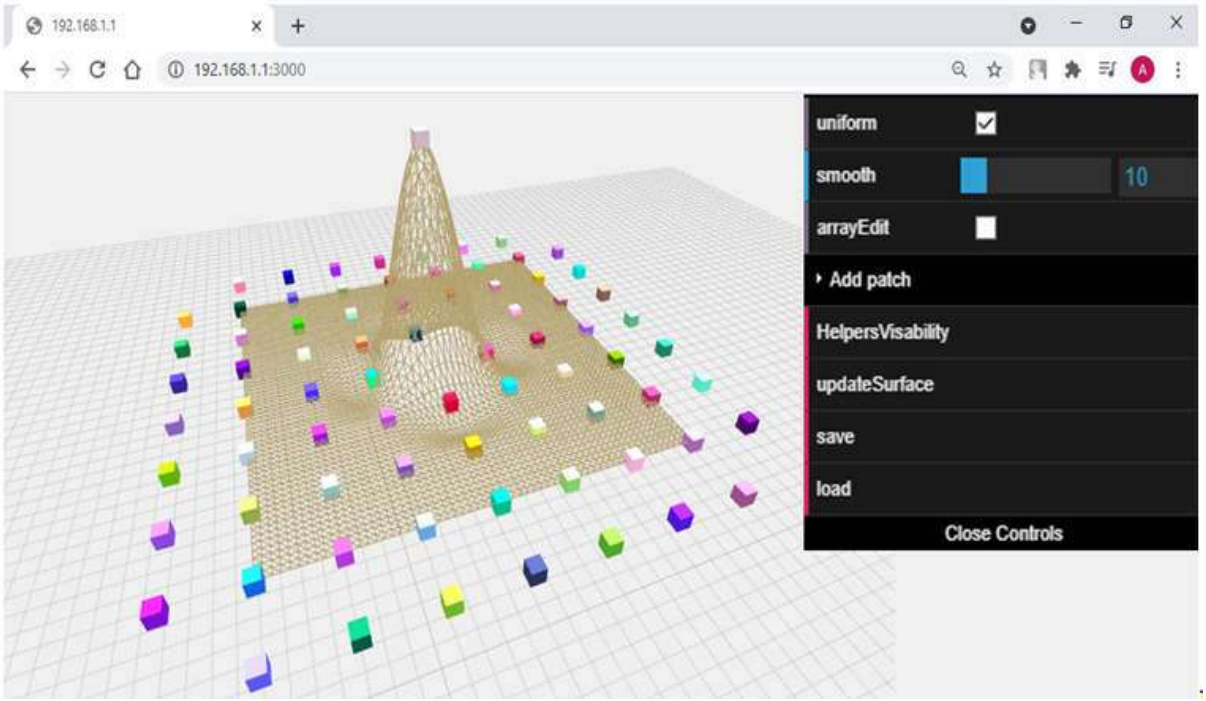

Рисунок 8 – Вигляд графічного інтерфейсу веб-застосунку, реалізованого за архітектурою SPA

Висновки. В роботі отримано алгоритмічну модель побудови поверхні Ерміта з умовою Кетмала-Рома та ненульовими векторами кручення поверхні. Показано, що умова Кетмала-Рома дає можливість склеювати окремі патчі з гладкістю першого порядку, що є запорукою ергономічності поверхонь. Показано, що окремо взята контрольна точка має локальний вплив на поверхню, а саме на 12 навколишніх патчей.

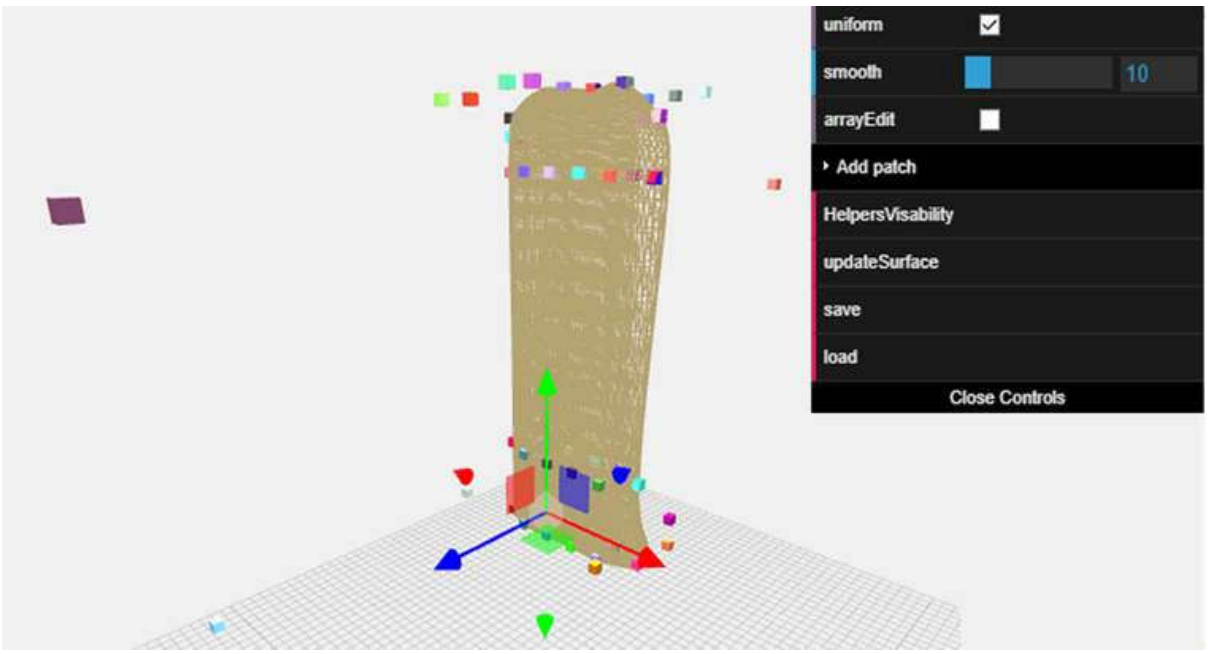

Рисунок 9 –Модель оболонки смартфону, яка представлена
за допомогою 30 патчів поверхні Кетмала-Рома

Розробка програмної системи моделювання поверхонь об'єктів з клієнтською частиною у вигляді веб-застосунку, який побудовано на основі архітектурного стилю SPA, показала, що досвід користувача такого застосунку близький до досвіду користування десктопною програмою. В той же самий час SPA застосунок не потребує інсталяції та може з успіхом виконуватись, як на стаціонарних, так і на мобільних пристроях.

REFERENCES

1. Yamaguchi F. Curves and Surfaces in Computer Aided Geometric Design / F. Yamaguchi. – Springer, 2014. – 396 p.

2. Salomon D. The Computer Graphics Manual / D. Salomon. – Springer, 2011. – 1526 p.

3. Agoston Max K. Computer Graphics and Geometric Modelling / Max K. Agoston. Springer, 2005. – P. 524.

4. Tickoo S. Autodesk 3ds Max 2019: A Comrehensive Guide / S. Tickoo. – CADCIM Technologies, 2018. – 720 p.

5. Skolski P. Single-page application vs. multiple-page application / P. Skolski – Source: https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58

6. Rogers D., Adams J. A. Mathematical Elements for Computer Graphics / D. Rogers, J. A. Adams. McGraw-Hill Science/Engineering/Math, 1989. – 512p.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

7. Farin G. Curves and Surfaces for CAGD: A Practical Guide / G. Farin. Morgan Kaufmann, 2001. – P. 291.

8. Zhou Xi-jun G2 Continuity Algorithms between Adjacent NURBS Patches along Common Cubic Boundary Curve / Xi-jun Zhou. - Chinese Journal of Aeronautics 16(4), 2003. – P. 241-246. DOI: 10.1016/S1000-9361(11)60191-X

9. Geometric continuity constraints for adjacent NURBS patches in shape optimisation / X. Zhang, Y. Wang, M. Gugala, J.-D. Muller. ECCOMAS Congress 2016. – P 1-14. DOI:10.7712/100016.2086.9316

10. Abramowitz M., Stegun I. Handbook of Mathematical Functions: with Formulas, Graphs, and Mathematical Tables M. Abramowitz, I. A. Stegun. Dover Publications, 1972. – P. 884.

Received 13.10.2021.
Accepted 15.10.2021.

Web-system for modeling surfaces based on Catmull-Rom patches

Today, surfaces play an important role in the work of designers, scientists, artists, surgeons and other professionals involved in creating innovative products. Development of utensils, furniture, automobile chassis, phones, clothes, buildings, even human bodies involves geometrical modeling of surfaces. NURBS modeling is the technology of non-uniform rational B-splines creating smooth forms and models that have no sharp edges. The characteristic makes NURBS as the analytical model of choice in Autodesk 3ds Max, Blender, Autodesk Maya, ZBrush, and other modeling systems. A generalizing characteristic of the given software systems is the use of a monolithic architectural style of software development, which is typical for desktop applications. Desktop software systems require installation on a local computer, which in turn binds the user to a specific operating system. NURBS accurately describe conical surfaces. Although each control has its own weight, and each weight has a local effect on the surface, a change in the weight of one vertex leads to a change in the entire surface.

One of the main requirements for 3D object modeling software is the ability to change the shape of the surface freely as it passes through all control points. An algorithmic model of the Hermit surface construction under the Catmull-Rom condition and nonzero surface torsion vectors is obtained. It is shown that the Catmull-Rom condition makes it possible to glue individual patches with first-order smoothness, which is a guarantee of ergonomic surfaces. It is shown that a single control point has a local effect on the surface, namely on the 12 surrounding patches. The development of a software system for modeling the surfaces of objects with the client part in the form of a web application, which is based on the architectural style of SPA, showed that the user experience of such an application is close to the experience of using a desktop program. At the same time, the SPA application does not require installation and successfully runs on both stationary and mobile devices.

Веб-система моделирования поверхностей на основе патчей Каттмалла-Рома

В работе получена алгоритмическая модель построения поверхности Эрмита с условием Каттмалла-Рома и ненулевыми векторами вращения поверхности. Показано, что условие Каттмалла-Рома дает возможность склеивать отдельные патчи с гладкостью первого порядка, являющимся залогом эргономики поверхностей. Показано, что отдельно взятая контрольная точка оказывает локальное влияние на поверхность, а именно на 12 окружающих патчей. Разработка программной системы моделирования поверхностей объектов с клиентской частью в виде веб-приложения, построенного на основе архитектурного стиля SPA, показала, что опыт пользователя такого приложения близок к опыту использования десктопной программы. В то же время SPA приложение не требует установки и может с успехом выполняться как на стационарных, так и на мобильных устройствах.

Демчишин Анатолій Анатолійович – доцент кафедри автоматизації проектування енергетичних процесів та систем, Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського".

Бурєнков Юрій Станіславович – магістрант кафедри автоматизації проектування енергетичних процесів та систем, Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського".

Демчишин Анатолий Анатольевич – доцент кафедры автоматизации проектирования энергетических процессов и систем, Национальный технический университет Украины "Киевский политехнический институт имени Игоря Сикорского".

Буренков Юрий Станиславович – магистрант кафедры автоматизации проектирования энергетических процессов и систем, Национальный технический университет Украины "Киевский политехнический институт имени Игоря Сикорского".

Demchyshyn Anatoliy – assistant professor, department of automation of design of energy processes and systems, National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”.

Burienkov Yurii – graduate student of the department of automation of design of energy processes and systems, National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)