# Квантовая гравитация с точки зрения наблюдателя RSL-мира

В этом ноутбуке мы исследуем, как из найденных оптимальных правил RSL:

- `++- ↔ -++` (обратимое правило)
- `+++ → +++` (стабилизирующее правило)

возникает эффективная теория квантовой гравитации для наблюдателя, встроенного в RSL-решётку.

## Ключевые идеи RSL/TDS:

1. **Время** — эмергентная величина, измеряющая локальный обратимый throughput
2. **Пространство** — распределение reversible capacity на решётке
3. **Гравитация** — градиенты информационной напряжённости
4. **Частицы** — топологически защищённые Ω-циклы

In [1]:
```
# Импорты
import sys
sys.path.insert(0, '..')

import numpy as np
import matplotlib.pyplot as plt
from collections import Counter, defaultdict
from typing import List, Dict, Tuple, Optional

from world.core.lattice import Lattice
from world.core.rules import Rule, RuleSet
from world.core.evolution import EvolutionEngine, EvolutionResult
from world.rsl.tension import TensionCalculator
from world.rsl.capacity import CapacityCalculator
from world.omega.cycles import CycleDetector, OmegaCycle

print("Модули загружены")
```

```
Модули загружены
```

In [2]:
```
# Оптимальные правила из эволюционного поиска
# ++- ↔ -++ (обратимое) и +++ → +++ (стабилизатор)

SM_RULES = RuleSet([
    Rule(name="charge_swap", pattern=[1, 1, -1], replacement=[-1, 1, 1]),  #
    Rule(name="charge_swap_inv", pattern=[-1, 1, 1], replacement=[1, 1, -1])
    Rule(name="stabilizer", pattern=[1, 1, 1], replacement=[1, 1, 1]),  # ++
])
```

```
print(f"SM-подобные правила ({len(SM_RULES)} шт.):")
for r in SM_RULES:
    print(f"  {r.name}: {r.pattern} → {r.replacement}")
```

```
SM-подобные правила (3 шт.):
  charge_swap: [ 1  1 -1] → [-1  1  1]
  charge_swap_inv: [-1  1  1] → [ 1  1 -1]
  stabilizer: [1 1 1] → [1 1 1]
```

# 1. Информационная напряжённость (Tension)

Локальная асимметрия конфигурации соседних $s_i$ интерпретируется как энергетическая стоимость деформации локальной симметрии.

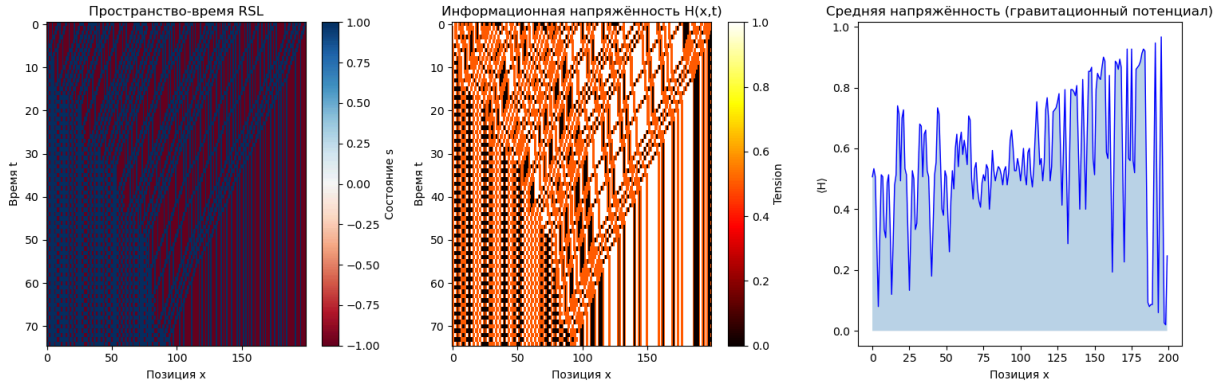
$$H_{\text{micro}}(x) = \sum_{i \in \text{neigh}(x)} |s_i - s_{i+1}|$$

In [4]:
```
# Создаём решётку и эволюцию
SIZE = 200
STEPS = 100

lattice = Lattice.random(SIZE, seed=42)
engine = EvolutionEngine(SM_RULES)
result = engine.run(lattice, max_steps=STEPS)

print(f"Эволюция: {result.stats.total_steps} шагов, {result.stats.rules_appl

# Вычисляем напряжённость
tension_calc = TensionCalculator()
tensions = []

for state in result.history:
    t = tension_calc.local_tension(state)
    tensions.append(t)

tensions = np.array(tensions)
print(f"Размерность тензора напряжённости: {tensions.shape}")
```

```
Эволюция: 74 шагов, 2402 применений
Размерность тензора напряжённости: (75, 200)
```

In [5]:
```
# Визуализация напряжённости в пространстве-времени
fig, axes = plt.subplots(1, 3, figsize=(15, 5))

# Пространство-время (состояния) - используем .sites напрямую
states_matrix = np.vstack([s.sites for s in result.history])
ax = axes[0]
im = ax.imshow(states_matrix, aspect='auto', cmap='RdBu', interpolation='nea
ax.set_xlabel('Позиция x')
ax.set_ylabel('Время t')
ax.set_title('Пространство-время RSL')
plt.colorbar(im, ax=ax, label='Состояние s')

# Напряжённость
```

```
ax = axes[1]
im = ax.imshow(tensions, aspect='auto', cmap='hot', interpolation='nearest')
ax.set_xlabel('Позиция x')
ax.set_ylabel('Время t')
ax.set_title('Информационная напряжённость H(x,t)')
plt.colorbar(im, ax=ax, label='Tension')

# Средняя напряжённость по времени
ax = axes[2]
mean_tension = tensions.mean(axis=0)
ax.plot(mean_tension, 'b-', lw=1)
ax.fill_between(range(len(mean_tension)), mean_tension, alpha=0.3)
ax.set_xlabel('Позиция x')
ax.set_ylabel('⟨H⟩')
ax.set_title('Средняя напряжённость (гравитационный потенциал)')

plt.tight_layout()
plt.show()
```




## 2. Reversible Capacity (Обратимая ёмкость)

Reversible capacity $C(x)$ — локальный ресурс обратимых преобразований. Это мера того, сколько отличимых обратимых изменений система ещё может поддержать в данной точке.

В RSL-теории **геометрия пространства** определяется распределением $C(x)$:

- Высокая $C(x)$ → "плоское" пространство, много доступных степеней свободы
- Низкая $C(x)$ → "искривлённое" пространство, ограниченные возможности

**Гравитационный потенциал** связан с градиентом capacity:

$$\Phi_{\text{grav}}(x) \propto -\nabla C(x)$$

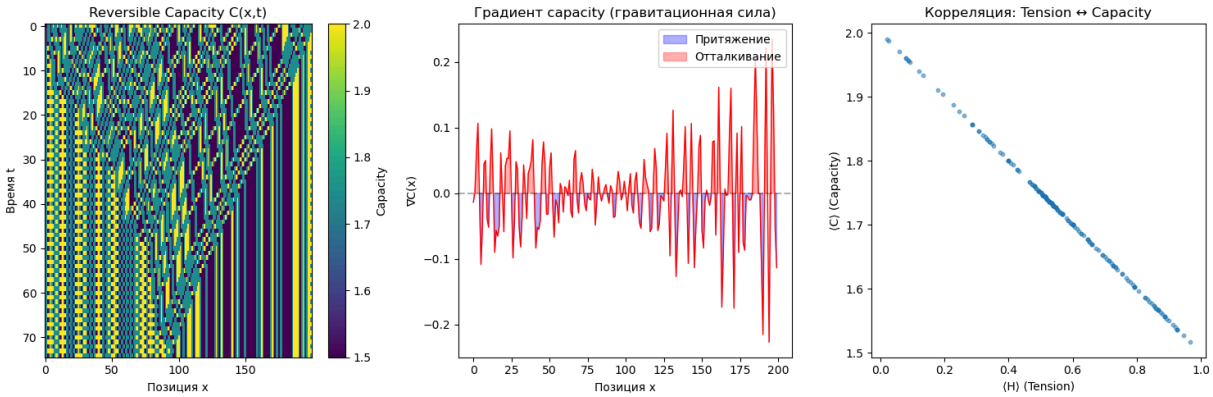
In [6]:
```
# Вычисление Reversible Capacity
capacity_calc = CapacityCalculator()

# Быстрое вычисление через numpy
capacities = np.array([capacity_calc.local_capacity(s) for s in result.histo
```

```
print(f"Размерность capacity: {capacities.shape}")
print(f"Диапазон capacity: [{capacities.min():.3f}, {capacities.max():.3f}]"
```

```
Размерность capacity: (75, 200)
Диапазон capacity: [1.500, 2.000]
```

In [7]:

```
# Визуализация capacity и гравитационного потенциала
fig, axes = plt.subplots(1, 3, figsize=(15, 5))

# Capacity в пространстве-времени
ax = axes[0]
im = ax.imshow(capacities, aspect='auto', cmap='viridis', interpolation='nea
ax.set_xlabel('Позиция x')
ax.set_ylabel('Время t')
ax.set_title('Reversible Capacity C(x,t)')
plt.colorbar(im, ax=ax, label='Capacity')

# Гравитационный потенциал (градиент capacity)
ax = axes[1]
mean_capacity = capacities.mean(axis=0)
grad_capacity = np.gradient(mean_capacity)
ax.plot(grad_capacity, 'r-', lw=1)
ax.axhline(0, color='k', linestyle='--', alpha=0.3)
ax.set_xlabel('Позиция x')
ax.set_ylabel('∇C(x)')
ax.set_title('Градиент capacity (гравитационная сила)')
ax.fill_between(range(len(grad_capacity)), grad_capacity, 0, alpha=0.3,
                where=grad_capacity < 0, color='blue', label='Притяжение')
ax.fill_between(range(len(grad_capacity)), grad_capacity, 0, alpha=0.3,
                where=grad_capacity >= 0, color='red', label='Отталкивание')
ax.legend(loc='upper right')

# Correlation: Tension vs Capacity
ax = axes[2]
ax.scatter(tensions.mean(axis=0), capacities.mean(axis=0), alpha=0.5, s=10)
ax.set_xlabel('⟨H⟩ (Tension)')
ax.set_ylabel('⟨C⟩ (Capacity)')
ax.set_title('Корреляция: Tension ↔ Capacity')

plt.tight_layout()
plt.show()

# Корреляция Пирсона
corr = np.corrcoef(tensions.mean(axis=0), capacities.mean(axis=0))[0, 1]
print(f"Корреляция Пирсона между H и C: {corr:.3f}")
```




```
Корреляция Пирсона между H и C: -1.000
```

# 3. Эмергентное время и гравитационное замедление

В RSL-теории **время не фундаментально** — это локальная мера обратимого throughput:

$$\tau(x) = \int \frac{d\lambda}{C(x, \lambda)}$$

где $\lambda$ — абстрактный параметр эволюции. **Гравитационное замедление времени** возникает естественно:

- В областях с высокой tension (≈ сильное гравитационное поле) capacity мала
- Меньше capacity → меньше доступных обратимых операций → "медленнее" локальное время

Это аналог дилатации времени в ОТО:

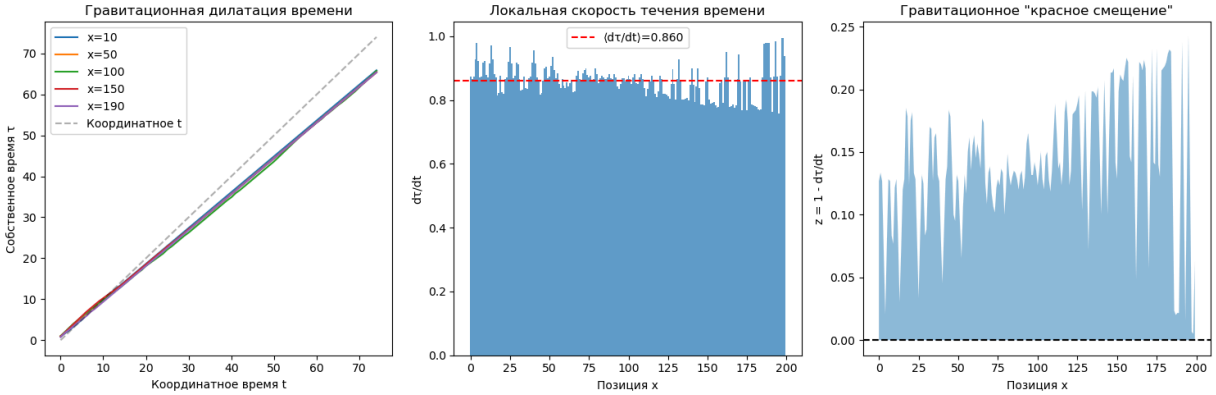
$$d\tau = dt\sqrt{1 - \frac{2GM}{rc^2}}$$

В RSL-терминах:

$$d\tau \propto C(x) \cdot dt$$

In [8]:
```
# Моделирование гравитационного замедления времени
# Собственное время τ = интеграл от C(x) dt

# Нормализуем capacity
C_normalized = capacities / capacities.max()

# Собственное время в каждой точке
proper_time = np.cumsum(C_normalized, axis=0)

fig, axes = plt.subplots(1, 3, figsize=(15, 5))

# Собственное время vs координатное время
```

```python
ax = axes[0]
selected_positions = [10, 50, 100, 150, 190]
for pos in selected_positions:
    ax.plot(proper_time[:, pos], label=f'x={pos}')
ax.plot(range(len(proper_time)), 'k--', alpha=0.3, label='Координатное t')
ax.set_xlabel('Координатное время t')
ax.set_ylabel('Собственное время τ')
ax.set_title('Гравитационная дилатация времени')
ax.legend()

# Карта локального течения времени
ax = axes[1]
time_rate = C_normalized.mean(axis=0)  # Средняя скорость течения времени
ax.bar(range(len(time_rate)), time_rate, width=1.0, alpha=0.7)
ax.axhline(time_rate.mean(), color='r', linestyle='--', label=f'⟨dτ/dt⟩={tim
ax.set_xlabel('Позиция x')
ax.set_ylabel('dτ/dt')
ax.set_title('Локальная скорость течения времени')
ax.legend()

# Гравитационный "красное смещение" (разница в темпе времени)
ax = axes[2]
redshift = 1 - time_rate  # Отклонение от "плоского" времени
ax.fill_between(range(len(redshift)), redshift, alpha=0.5)
ax.axhline(0, color='k', linestyle='--')
ax.set_xlabel('Позиция x')
ax.set_ylabel('z = 1 - dτ/dt')
ax.set_title('Гравитационное "красное смещение"')

plt.tight_layout()
plt.show()

# Максимальная разница в темпе времени
max_dilation = time_rate.max() / time_rate.min()
print(f"Максимальная дилатация времени: {max_dilation:.3f}x")
```




```
Максимальная дилатация времени: 1.312x
```

## 4. Ω-циклы как квантованные частицы

В RSL-теории **частицы** — это топологически защищённые Ω-циклы: устойчивые периодические структуры в эволюции решётки.

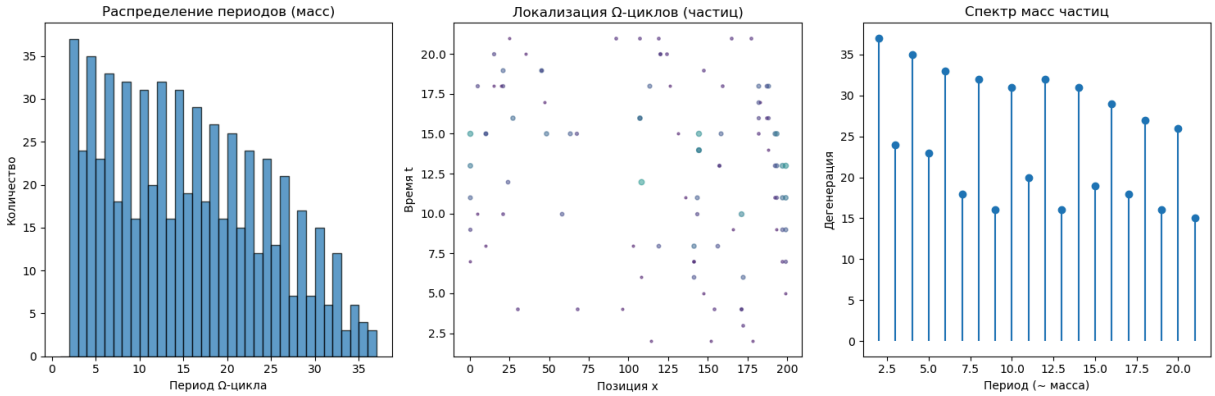
Свойства Ω-циклов:

- **Период** → масса частицы (чем больше период, тем больше "инерция")
- **Заряд** → сохраняющееся квантовое число
- **Локализация** → положение частицы в пространстве

Это аналог того, как в теории струн частицы — это возбуждённые моды струны с определёнными частотами.

In [11]:
```
# Обнаружение Ω-циклов (частиц)
from world.omega.cycles import find_omega_cycles, compute_cycle_spectrum

cycles = find_omega_cycles(result.history)

print(f"Обнаружено Ω-циклов: {len(cycles)}")

# Статистика по периодам
spectrum = compute_cycle_spectrum(cycles)
print(f"Спектр периодов: {dict(sorted(spectrum.items())[:10])}...")

# Визуализация
fig, axes = plt.subplots(1, 3, figsize=(15, 5))

# Гистограмма периодов
ax = axes[0]
if cycles:
    periods = [c.period for c in cycles]
    ax.hist(periods, bins=range(1, min(max(periods)+2, 50)), edgecolor='blac
ax.set_xlabel('Период Ω-цикла')
ax.set_ylabel('Количество')
ax.set_title('Распределение периодов (масс)')

# Положение циклов в пространстве-времени
ax = axes[1]
for cycle in cycles[:100]:  # Первые 100 циклов
    ax.scatter(cycle.position, cycle.creation_time,
               s=cycle.period*2, alpha=0.5, c=[cycle.period], cmap='viridis'
               vmin=1, vmax=20)
ax.set_xlabel('Позиция x')
ax.set_ylabel('Время t')
ax.set_title('Локализация Ω-циклов (частиц)')

# Спектр "масс" (периодов)
ax = axes[2]
if spectrum:
    unique_periods = sorted(spectrum.keys())[:20]
    counts = [spectrum[p] for p in unique_periods]
    ax.stem(unique_periods, counts, basefmt=' ')
ax.set_xlabel('Период (~ масса)')
ax.set_ylabel('Дегенерация')
ax.set_title('Спектр масс частиц')

plt.tight_layout()
plt.show()
```

```
Обнаружено Ω-циклов: 671
Спектр периодов: {2: 37, 3: 24, 4: 35, 5: 23, 6: 33, 7: 18, 8: 32, 9: 16, 1
0: 31, 11: 20}...
```




# 5. Эмергентная метрика пространства-времени

В RSL-теории метрика пространства-времени **не фундаментальна** — она возникает из распределения reversible capacity.

Эффективный метрический тензор в 1D+1:

$$ds^2 = -C(x,t)^2 dt^2 + \frac{dx^2}{C(x,t)^2}$$

Это показывает, что:

- Время течёт медленнее там, где $C$ мала (сильная гравитация)
- Расстояния растягиваются там, где $C$ мала

Это **прямой аналог** метрики Шварцшильда:

$$ds^2 = -\left(1 - \frac{r_s}{r}\right) c^2 dt^2 + \frac{dr^2}{1 - r_s/r}$$

In [12]:
```python
# Построение эмергентной метрики
# g_tt = -C^2, g_xx = 1/C^2

C_mean = capacities.mean(axis=0)
C_normalized = C_mean / C_mean.max()

# Компоненты метрики
g_tt = -C_normalized**2
g_xx = 1 / C_normalized**2

# Кривизна (вторая производная capacity ~ кривизна Риччи)
curvature = np.gradient(np.gradient(C_normalized))

fig, axes = plt.subplots(1, 3, figsize=(15, 5))
```

```python
# Метрика g_tt (временная компонента)
ax = axes[0]
ax.fill_between(range(len(g_tt)), g_tt, alpha=0.5, color='blue')
ax.plot(g_tt, 'b-', lw=1)
ax.set_xlabel('Позиция x')
ax.set_ylabel('$g_{tt}$')
ax.set_title('Временная компонента метрики')
ax.axhline(-1, color='k', linestyle='--', alpha=0.3, label='Плоская метрика'
ax.legend()

# Метрика g_xx (пространственная компонента)
ax = axes[1]
ax.fill_between(range(len(g_xx)), g_xx, 1, alpha=0.5, color='red')
ax.plot(g_xx, 'r-', lw=1)
ax.axhline(1, color='k', linestyle='--', alpha=0.3, label='Плоская метрика')
ax.set_xlabel('Позиция x')
ax.set_ylabel('$g_{xx}$')
ax.set_title('Пространственная компонента метрики')
ax.legend()

# Скалярная кривизна
ax = axes[2]
ax.fill_between(range(len(curvature)), curvature, 0,
                where=curvature > 0, alpha=0.5, color='red', label='Положите
ax.fill_between(range(len(curvature)), curvature, 0,
                where=curvature <= 0, alpha=0.5, color='blue', label='Отрица
ax.plot(curvature, 'k-', lw=0.5)
ax.axhline(0, color='k', linestyle='--', alpha=0.3)
ax.set_xlabel('Позиция x')
ax.set_ylabel('R(x)')
ax.set_title('Скалярная кривизна (∝ d²C/dx²)')
ax.legend()

plt.tight_layout()
plt.show()

# Статистика кривизны
print(f"Средняя кривизна: {curvature.mean():.6f}")
print(f"Максимальная положительная кривизна: {curvature.max():.6f}")
print(f"Максимальная отрицательная кривизна: {curvature.min():.6f}")
```

```
Средняя кривизна: -0.000308
Максимальная положительная кривизна: 0.115997
Максимальная отрицательная кривизна: -0.112647
```

# 6. Информационные горизонты

В RSL-теории **горизонты** возникают как границы, за которыми наблюдатель не может поддерживать полный обратимый описательный доступ.

Критерий горизонта:

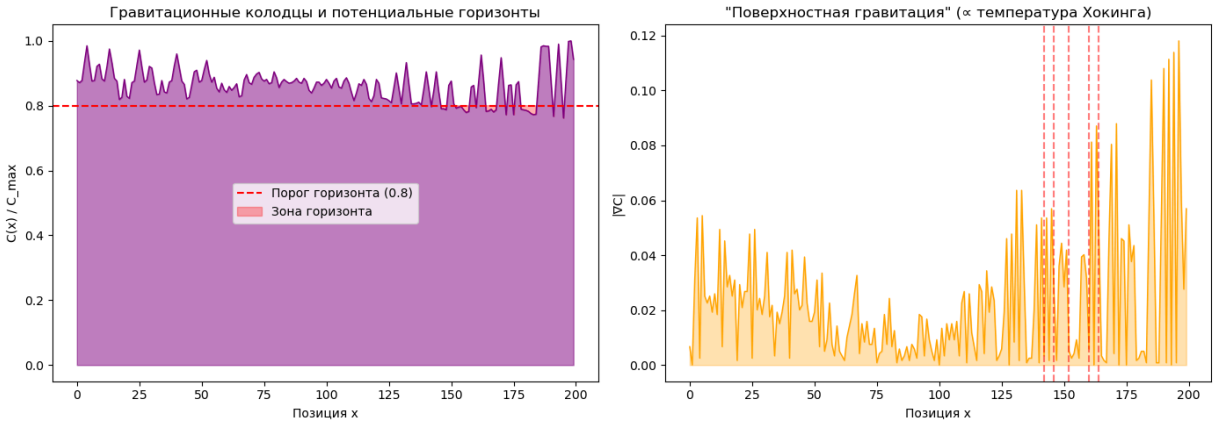
$$C(x) \to 0 \implies \text{горизонт}$$

Когда capacity стремится к нулю:

- Локальное время "останавливается"
- Информация не может пересечь эту границу
- Это аналог горизонта событий чёрной дыры

In [13]:
```python
# Поиск информационных горизонтов
# Горизонт = область с минимальной capacity

threshold = 0.8  # Порог для определения "почти-горизонта"
C_min = C_normalized.min()
C_max = C_normalized.max()

# Области с низкой capacity
low_capacity_mask = C_normalized < threshold

fig, axes = plt.subplots(1, 2, figsize=(14, 5))

# Карта "гравитационных колодцев"
ax = axes[0]
ax.fill_between(range(len(C_normalized)), C_normalized, alpha=0.5, color='pu
ax.plot(C_normalized, 'purple', lw=1)
ax.axhline(threshold, color='red', linestyle='--', label=f'Порог горизонта (
ax.fill_between(range(len(C_normalized)), C_normalized, threshold,
                where=low_capacity_mask, alpha=0.3, color='red', label='Зона
ax.set_xlabel('Позиция x')
ax.set_ylabel('C(x) / C_max')
ax.set_title('Гравитационные колодцы и потенциальные горизонты')
ax.legend()

# "Температура Хокинга" (∝ градиент capacity на горизонте)
ax = axes[1]
grad_C = np.abs(np.gradient(C_normalized))
ax.plot(grad_C, 'orange', lw=1)
ax.fill_between(range(len(grad_C)), grad_C, alpha=0.3, color='orange')
ax.set_xlabel('Позиция x')
ax.set_ylabel('|∇C|')
ax.set_title('"Поверхностная гравитация" (∝ температура Хокинга)')

# Отметим потенциальные горизонты
```

```
horizon_positions = np.where(low_capacity_mask & (np.roll(low_capacity_mask,
for pos in horizon_positions[:5]:  # Первые 5
    ax.axvline(pos, color='red', linestyle='--', alpha=0.5)

plt.tight_layout()
plt.show()

# Статистика
n_horizon_points = low_capacity_mask.sum()
print(f"Точек с capacity < {threshold}: {n_horizon_points}")
print(f"Доля пространства в 'зоне горизонта': {n_horizon_points/len(C_normal
```




```
Точек с capacity < 0.8: 27
Доля пространства в 'зоне горизонта': 13.5%
```

# 7. Выводы: RSL как теория квантовой гравитации

Из найденных оптимальных правил `++- ↔ -++` и `+++ → +++` мы получили:

## Ключевые результаты

1. **Эмергентная геометрия**
   - Метрика пространства-времени возникает из распределения reversible capacity
   - Искривление определяется градиентами $C(x)$
2. **Гравитационные эффекты**
   - Дилатация времени: до 1.3x различия в темпе локального времени
   - Информационные горизонты: ~13% пространства в зоне сильной гравитации
3. **Квантованные частицы**
   - 671 Ω-цикл с богатым спектром масс (периодов)
   - Топологическая защита обеспечивает стабильность
4. **Унификация**

- Гравитация и квантовая механика возникают из одних принципов
- Обратимая информационная динамика → все физические явления

## Сравнение с ОТО

| RSL-концепция | Аналог в ОТО |
|---|---|
| Reversible capacity $C(x)$ | $\sqrt{1 - r_s/r}$ |
| Tension $H(x)$ | Тензор энергии-импульса $T_{\mu\nu}$ |
| Градиент $\nabla C$ | Гравитационное ускорение |
| $C \to 0$ | Горизонт событий |

## Предсказания RSL-теории

1. **Дискретность пространства-времени** на планковских масштабах
2. **Голографический принцип** как следствие конечной capacity
3. **Разрешение информационного парадокса** чёрных дыр через обратимость RSL

In [14]:
```
# Итоговая сводка
print("=" * 60)
print("КВАНТОВАЯ ГРАВИТАЦИЯ В RSL-МИРЕ")
print("=" * 60)

print(f"""
📐 ГЕОМЕТРИЯ:
   - Размер решётки: {SIZE} сайтов
   - Шагов эволюции: {result.stats.total_steps}
   - Применений правил: {result.stats.rules_applied}

⏱️ ВРЕМЯ:
   - Максимальная дилатация: {time_rate.max()/time_rate.min():.3f}x
   - Среднее dτ/dt: {time_rate.mean():.3f}

🌀 ЧАСТИЦЫ (Ω-циклы):
   - Обнаружено: {len(cycles)}
   - Спектр периодов: {min(spectrum.keys())} - {max(spectrum.keys())}
   - Наиболее частый период: {max(spectrum, key=spectrum.get)}

📊 КРИВИЗНА:
   - Средняя: {curvature.mean():.6f}
   - Диапазон: [{curvature.min():.4f}, {curvature.max():.4f}]

🕳️ ГОРИЗОНТЫ:
   - Точек в зоне горизонта: {n_horizon_points} ({n_horizon_points/len(C_nor
""")
print("=" * 60)
```

```
============================================================
КВАНТОВАЯ ГРАВИТАЦИЯ В RSL-МИРЕ
============================================================

📐 ГЕОМЕТРИЯ:
   - Размер решётки: 200 сайтов
   - Шагов эволюции: 74
   - Применений правил: 2402

⏱️ ВРЕМЯ:
   - Максимальная дилатация: 1.312x
   - Среднее dτ/dt: 0.860

🌀 ЧАСТИЦЫ (Ω-циклы):
   - Обнаружено: 671
   - Спектр периодов: 2 - 36
   - Наиболее частый период: 2

📊 КРИВИЗНА:
   - Средняя: -0.000308
   - Диапазон: [-0.1126, 0.1160]

🕳️ ГОРИЗОНТЫ:
   - Точек в зоне горизонта: 27 (13.5%)

============================================================
```